D6.4

USEMP disclosure scoring framework and disclosure setting framework – v2

V1.1 2016-02-15

Georgios Petkos (CERTH), Symeon Papadopoulos (CERTH), Juxhin Bakalli (CERTH), Eleftherios Spiromitros-Xioufis (CERTH), Theodoros Michalareas (VELTI), Andreas Drakos (VELTI), Yiannis Kompatsiaris (CERTH)

This deliverable provides an update on the activities carried out within tasks T6.1 and T6.2, between the submission of D6.1-D6.2 and the end of the second year of the project. The most important activity in this period has been the implementation of the disclosure scoring and setting frameworks, the designs of which were presented in D6.1 and D6.2 respectively. In terms of disclosure scoring, a range of methods have been examined for predicting user attributes and extensive experiments have been carried out on data collected during the preliminary pilot testing of the integrated prototype. Additionally, related to the development of the disclosure settings framework, a novel approach that performs automatic grouping of the friends of an OSN user has also been developed. Moreover, the work on trackers and Do Not Track policies has progressed by implementing a number of enhancements on the DataBait plug-in and by examining mechanisms for recommending trackers to be blocked.

Project acronym	USEMP
Full title	User Empowerment for Enhanced Online Presence Management
Grant agreement number	611596
Funding scheme	Specific Targeted Research Project (STREP)
Work program topic	Objective ICT-2013.1.7 Future Internet Research Experimentation
Project start date	2013-10-01
Project Duration	36 months

Workpackage 1	User Assistance for Shared Personal Data Management
Deliverable lead org.	CERTH
Deliverable type	Prototype
Authors	Georgios Petkos (CERTH) Symeon Papadopoulos (CERTH) Juxhin Bakalli (CERTH) Eleftherios Spiromitros-Xioufis (CERTH) Theodoros Michalareas (VELTI) Andreas Drakos (VELTI) Yiannis Kompatsiaris (CERTH)
Reviewers	Noel Catterall (HWC) Laurence Claeys (iMinds)
Version	1.1
Status	Final
Dissemination level	PU: Public
Due date	2015-09-30
Delivery date	2015-10-29 (revised 2016-02-15)

Version	Changes
0.1	First outline ToC by Georgios Petkos and Symeon Papadopoulos
0.2	Content filled in chapters 4 and 6 by Andreas Drakos and VELTI
0.3	First versions of chapters 2, 3 and 5 by Georgios Petkos
0.4	Revised versions of chapters 4 and 6 and Annex III inserted by Andreas Drakos and VELTI

0.5	Merged chapters 4 and 5 and updates on chapters 3 and 6 by Giorgos Petkos and Eleftherios Spiromitros-Xioufis
0.6	Various updates and completions, first complete draft by Georgios Petkos and Symeon Papadopoulos
0.7	Various updates and refinements by Symeon Papadopoulos
0.8	First complete draft, sent for internal review by Georgios Petkos
0.9	Updates after comments received from the reviewers
1.0	Final refinements and fixes
1.1	Updated version addressing comments from the second annual review

Table of Contents

1. Introduction

This deliverable is an update of deliverables D6.1 (USEMP privacy scoring framework – v1, associated with task T6.1) and D6.2 (USEMP privacy setting framework – v1, associated with task T6.2). It reports on work carried out within these tasks between the submission of D6.1 (M15) and D6.2 (M16) and the end of the second year of the project (M24). It should be noted that according to the original DoW, D6.4 would be an update of D6.1 only. Nevertheless, according to the recent amendment of the DoW the scope of D6.4 has been redefined as being an update of both D6.1 and D6.2. A further update will eventually come with D6.5. There are four main activities on which this deliverable reports: a) the development of the USEMP disclosure scoring framework, b) the development of a number of mechanisms for inferring personal user attributes, c) the development of the USEMP disclosure settings framework, d) the development of a tool that supports disclosure control with respect to the browsing behaviour of users via Web trackers and Do-Not-Track (DNT) policies. Work in the first two directions is carried out within task T6.1, whereas work in the latter two directions is carried out within task T6.2.

1.1. Overview of the deliverable

This deliverable is structured as follows. The next chapter examines the implementation of the disclosure scoring framework, the design of which was originally presented in D6.1. Then, in chapter 3 the user attribute inference mechanisms that were initially presented in D6.1 are revisited and tested using actual data collected during the USEMP early pilot testing of the integrated prototype. Importantly, different types of features extracted from the data, different types of classifiers and fusion strategies are examined and compared. Chapter 4 presents the implementation of the disclosure settings framework. Chapter 5 looks at Web trackers and DNT policies, with the aim of assisting the control of disclosed information related to the Web browsing behaviour of users. Chapters 2 and 3 correspond to work carried out within task T6.1, while chapters 4 and 5 to work carried out within task T6.2.

1.2. Privacy vs. disclosure

Initially, D6.1 introduced and used the term “Privacy scoring framework” and accordingly D6.2 the term “Privacy settings framework”. However, concerns were raised about the legal meaning of the term “privacy”. In particular, “privacy” in the legal sense in EU law refers to Article 8 of the European Convention of Human Rights[1] (right to respect for private life), which has very specific meaning (mainly a right that prevents power imbalances between state and citizen or citizen and other actors). Yet, the “privacy scoring framework” of D6.1 also contains information that does not fit this definition. For instance, the framework could represent the fact that a person’s OSN data indicates that he/she likes swimming, pets or some specific brand; this does not fit the definition of privacy in the legal (human right) sense. Nevertheless, the inclusion of such information in the scoring framework informs the user about the types of information that he/she discloses through his/her digital trails. This

[1] http://www.echr.coe.int/Documents/Convention_ENG.pdf

can be interesting in terms of transparency and data protection, but data protection is not equivalent to privacy. Therefore, it was decided to use the more accurate term “disclosure scoring framework” in lieu of “privacy scoring framework” and, accordingly, “disclosure settings framework” instead of “privacy settings framework” in the remainder of the project.

1.3. Main achievements

The main achievements during the reporting period include the following:

- The disclosure scoring framework was implemented. In addition, its implementation was made publicly available and the approach was published (Petkos et al., 2015a). Moreover, the first steps were made towards integrating it into DataBait (Chapter 2).
- The OSN and questionnaire data that were collected during the early pilot phase were used to train and test numerous classifiers that predict a variety of user attributes. A variety of features extracted from the collected OSN data, learning algorithms (including multi-target classification methods) and fusion mechanisms were evaluated (Chapter 3), and valuable insights were gained that will enable the enrichment of DataBait with user attribute prediction capabilities.
- Core facilities of the disclosure settings framework were implemented (Chapter 4).
- A novel probabilistic approach for automatically grouping the friends of an OSN user was developed. Importantly, the approach takes into account both the connectivity between friends and their profile properties. This led to the publication of a paper (Petkos et al., 2015b) (Chapter 4).
- Work was carried out on increasing OSN users’ awareness via a) examining and comparing a number of influence metrics, and b) experimenting with recommendation algorithms for building a content consumption model of users (Chapter 4).
- Previous work on assisting users to perceive and control disclosure of their browsing behaviour information by examining trackers and Do-Not-Track (DNT) policies was extended: apart from integrating the trackers tool to DataBait, a mechanism was examined for recommending to users a set of Web trackers to block (Chapter 5).

2.Overview of Disclosure Scoring Framework

2.1. Review of disclosure scoring framework

One of the main goals of WP6 is to raise the awareness of OSN users with respect to their OSN presence, i.e. to enhance their perception about the types of personal information that is disclosed to the OSN platform. To this end, in D6.1 a privacy scoring framework was presented (henceforth renamed to “disclosure scoring framework”).

In the period since the delivery of D6.1, activities with a focus on the implementation of the disclosure scoring framework were carried out. Details about the implementation will be presented in the next section. For convenience, the structure of the disclosure scoring framework is briefly revisited first.

The development of the scoring framework was based on three requirements that were derived from existing user studies (Madejski et al., 2012; Knijneburg et al., 2013; Knijneburg, 2014) and the defined social requirements in D4.2:

1. The framework should take into account the fact that privacy concerns are likely to differ between users, and therefore it is important to consider each user’s personal preferences in order to compute privacy scores.
2. Different types of information have different significance to users, and therefore the framework should be structured according to the different types of information.
3. The framework should also take into account inferred information, and should be generic enough so that different inference mechanisms can be added to it and extend it in a seamless manner. Also, ideally, it should be able to link the inferences made to the specific OSN presence data that support the particular inference.

In D6.1, different types of personal attributes (e.g. age, sexual orientation, political beliefs, etc.) were identified and organized in a set of categories or *dimensions* (e.g. demographics, consumer profile, health profile, etc.). It was also recognized that each attribute can take a number of values. Eventually, some values are linked to specific OSN data that “support” these values, i.e. they indicate that the value is valid for some user. The linking between the values and the data is performed by a number of inference and data analysis mechanisms. Eventually, this generates a hierarchical structure, organized in a semantic manner, consisting of a set of dimensions, attributes, values and links to data. This structure is called the *disclosure dimensions framework* (previously privacy dimensions framework).

On top of this structure, a number of scores are assigned to each node of the hierarchy (dimension, attribute, value). Different sets of scores are maintained for the nodes at each level of the hierarchy. The hierarchical structure of the framework and the scores that can be found at each level are depicted in Figure 1. These scores express different aspects of information disclosure. For instance, there are separate scores for visibility, sensitivity, level of user control over the disclosure, as well as an overall disclosure score. Importantly, at the values level, there are additional fields (non-scores) to provide valuable information to users. For instance, the support field of some value links it to particular OSN data associated with the value. This is important as it allows the user to link particular aspects of his/her disclosed information to particular content that he/she shared on the OSN.

The framework fits the above three requirements in the following ways (each item in the following list corresponds to an item in the list of requirements above):

1. The sensitivity of each type of information can be set by the user. This allows directly taking into account the different concerns that users may have about different types of information.
2. Different types of information are organized in a semantic manner.
3. The framework can take into account both declared and inferred information. The implementation is flexible and may accommodate different inference mechanisms.

For more details on the framework, please refer to D6.1 and (Petkos et al., 2015a).

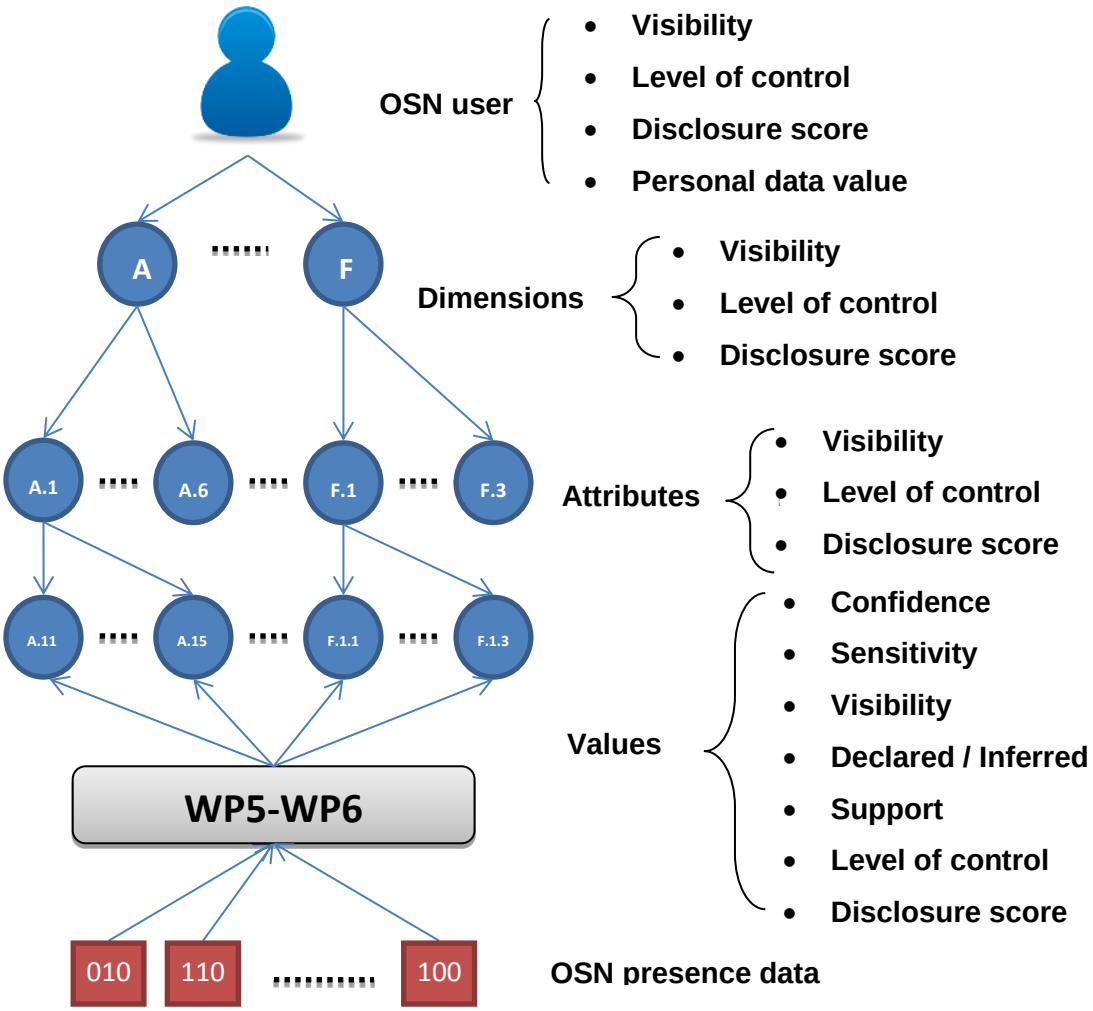


Figure 1. Overview of disclosure scoring framework

2.2. Overview of implementation

The implementation of the privacy scoring framework was completed and its source code was made publicly available and is maintained on GitHub[2]. The framework is implemented in Python and a Web interface is additionally provided. It uses a mongo database to store the privacy scores of users, a database that was referred to in previous deliverables (e.g. D7.1) as the "privacy database".

During the implementation, three different sources of information were considered to provide input to the disclosure scoring framework:

- Outputs produced by the analysis and inference modules of WP5 and WP6. Each such output populates the support field of some *values*. Based on the support records that a value may have (could be more than one, coming from different modules), additional fields are automatically derived using the computation mechanisms described in D6.1 (e.g., confidence, declared/inferred and disclosure score).
- Sensitivity is directly provided by users, based either on their own preferences or the preferences of a general population of users (based on estimates). As mentioned this covers the first design requirement of the scoring framework.
- Some fields are computed based on the properties of raw OSN data. For instance, the sharing settings of data that are pointed to by the support records are used to compute the level of control and visibility.

These three sources of information led to the design and services that the disclosure scoring framework supports. In particular, the framework provides the following endpoints:

- `/user/value/support`. This allows access to the raw data records that provide support for some value referring to a particular user. In terms of implementation, the support object is positioned one level below the values level in the hierarchy (not depicted in Figure 1). It is the link between the values level and the OSN data. This endpoint triggers the computation of scores and it is called by the backend process once results from the inference/analysis mechanisms are produced. This endpoint corresponds to the first item of the list of inputs to the disclosure scoring framework.
- `/user/value/sensitivity`, `/user/attribute/sensitivity` and `/user/dimension/ sensitivity`. These endpoints give the possibility to set the sensitivity at the dimension, attribute or value level. When the sensitivity is set on a level higher than the value level, the same sensitivity is set on all the levels below it in a cascade scheme. This corresponds to the second point from the list above.

The third source of information is addressed by directly retrieving data from the DataBait database that stores OSN data. That is, the implementation directly queries the OSN database with some content id in order to obtain the required information.

Additionally, the framework provides the following retrieval endpoints:

- `/users/list`. This endpoint will return the list of all users in the privacy database.
- `/user`. This call offers the possibility to create and delete users with all their respective data. It also allows obtaining the scores of some particular user.

[2] https://github.com/MKLab-ITI/usemp-pscore.

- `/user/value`. The implementation does not have the structure of the dimensions framework hardcoded in it. The reason for offering such an endpoint is to make the framework as flexible as possible. This function essentially creates the dimensions framework structure. It takes as parameter a path from the root of the hierarchy up to some value and builds the branch of the hierarchy that is missing to create the path. Initially, the dimensions structure is empty for each user and is progressively filled as more inferences related to different attributes come from the analysis modules. For instance, if the parameter of the first call was "demographics/gender/male" and the parameter of the second call was "religious_views/supported_religion/atheist", then after these first two calls only the two dimensions "demographics" and "religious views" would be defined for the specific user. Full CRUD (Create, Read, Update, Delete) capabilities are provided for these objects.

Importantly, the whole computation process is triggered when some new support object is produced by some inference/analysis module of WP5 or WP6. Data is automatically stored in the privacy database and is retrieved on demand – either by the scoring framework or other modules – using the appropriate endpoint. For instance, the scoring data for a user may be retrieved from the user interface (in the JSON format described in Annex II of D6.1) in order to visualize the different scores, dimensions, etc.

Also, it should be mentioned that the set of dimensions and attributes listed in Annex I of D6.1 were updated, following the final decisions about the attributes that were included in the questionnaire used in the early pilot tests. The latest version of the dimensions framework is presented in Annex II of this document.

The scoring framework is currently in the process of being integrated into the DataBait tool. The following figures provide preliminary visualizations of the scoring framework using the bubbles representation that was presented in D6.3. Please note that there are aesthetic and organizational differences between this preliminary implementation of the visualization and the original design that was presented in D6.3. Some of the differences, e.g. the spatial arrangement of the bubbles after some dimension has been selected, have been informed by more recent developments, in particular an iterative evaluation and design process that has been carried out and that will be presented in D6.6. Please also note that the actual deployment of the scoring framework and this visualization in DataBait is still pending.

Figure 2 shows the initial screen of the visualization of the scoring framework[3]. This includes nodes corresponding to the nine disclosure dimensions. At all levels of the hierarchy, the size of a node is linked to the overall disclosure of the node (dimension, attribute or value) and its colour is linked to the sensitivity of the node.

Figure 3 shows the visualization once a dimension has been selected: the dimensions are placed on the left of the screen and the attributes under the selected dimension are shown. Finally, Figure 4 shows the visualization when the user drills down to the values level. Importantly, whenever the user hovers over some node, the relevant scoring data are shown on the right of the screen. When the user hovers over a node at the values level, then the OSN data that is included in the support of the value are shown. This will allow the user to perceive the link between specific data that the user has shared and the disclosure of

[3] Also available for demo purposes at: http://usemp-mklab.iti.gr/usemp/. Please note that the data shown here and at the demo are completely random.

specific types of information. Additionally, in order to assist the user to perceive potential risks entailed by the disclosure of different types of information, the interface will eventually also list a number of potential threats that may be linked with the disclosure of specific types of information. For instance, the disclosure of specific demographics attributes may be associated with discriminatory behaviour in specific scenarios and the user will be notified about such cases.

Figure 2. Initial screen of the disclosure scoring visualization: The nine disclosure dimensions are shown at the top level. In this instance, the size of the nodes denotes the overall disclosure score of the node and the color denotes the sensitivity of the node: red nodes have high sensitivity, whereas yellow nodes have medium sensitivity.

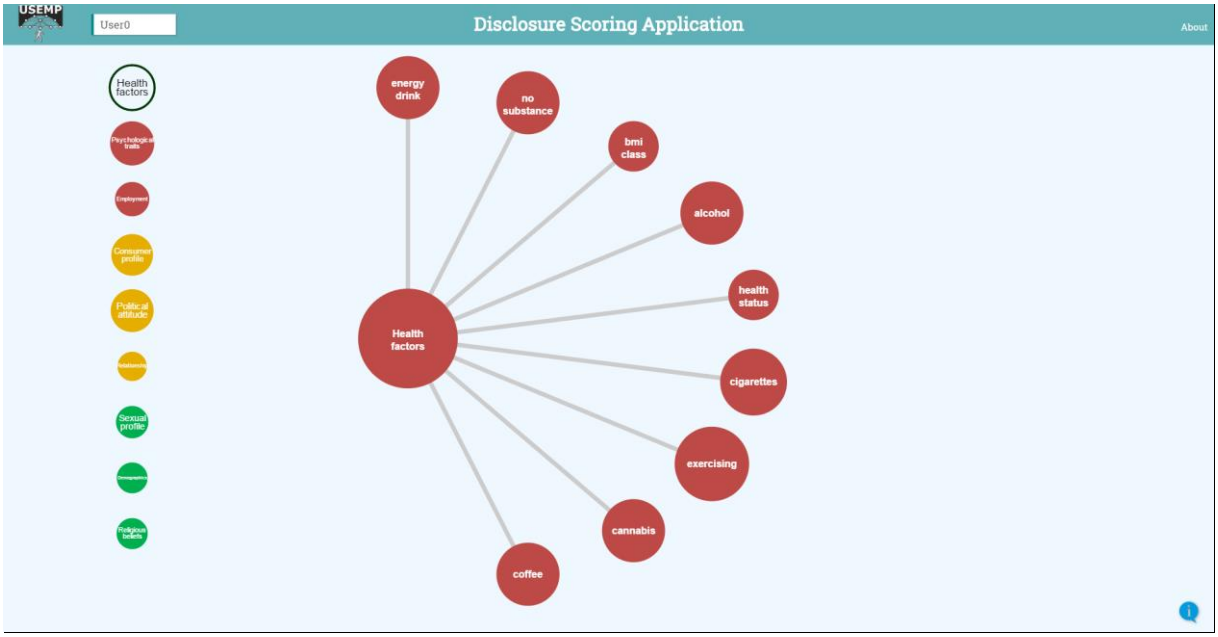


Figure 3. Disclosure scoring visualization: once a dimension has been selected, the attributes under it are also shown.

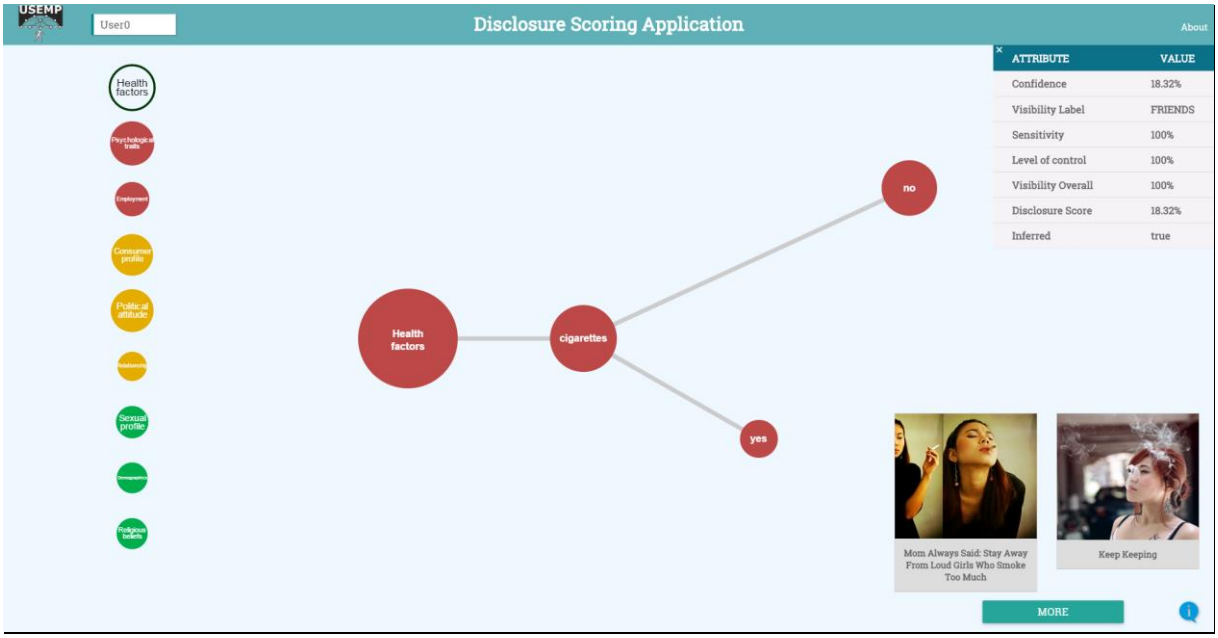


Figure 4. Disclosure scoring visualization: the user can drill down to the values level. If the user hovers over some node, then the scoring parameters of the node are shown to the right of the screen. Additionally, if the user clicks on some node at the values level, the data that is included in the support of this value is also displayed, helping the user understand how this value is linked to the data that the user shared.

3.Predicting Personal Information Disclosure

Work in task T6.1 involves not only the development of the disclosure scoring framework, but also the development of inference mechanisms that feed into it. In D6.1, three inference mechanisms were proposed and tested based on data external to USEMP. In particular:

1. based on users' OSN trails (in particular their "likes" on Facebook);
2. based on topics extracted from the textual content that users post and the textual content of Facebook pages that users like;
3. based on the connectivity structure between the user and his/her friends and the characteristics of his/her friends.

Different variations were examined for each of these methods in D6.1. For instance, for the first method, different classifiers and feature selection approaches were tested. These three inference mechanisms were tested using data either from the My Personality dataset (for the first two) or data collected from Twitter (for the third).

Following the delivery of D6.1, relevant data that can be used to test and extend these methods were collected during the early pilot tests of the DataBait tool. More specifically, during these tests, the OSN data of 170 Facebook users were obtained and at the same time these users were asked to fill in a questionnaire to directly provide information regarding their attributes (i.e. ground truth). This data could then be used to both train and test the inference modules. Unfortunately, it was not possible to obtain data about the friends of users at this stage and therefore the third, connectivity based mechanism, could not be assessed with USEMP data. As a result, comprehensive tests were conducted on the first two mechanisms and new extensions were proposed. Details about these experiments and results are provided below.

3.1. Data and experimental setup

The OSN data collected during the early pilots include the following types of data: a) Likes, b) Textual posts and c) Images. It is clear that different types of features can be extracted from this data. For instance, direct use of likes corresponds to the first inference mechanism listed above and the use of topics extracted from textual posts and likes corresponds to the second. Nevertheless, it is possible to extract other features as well. Therefore, in the following, different types of features (in addition to those presented in D6.1) are considered. In all cases, the extracted features are fed into different types of classifier. In the remainder of this chapter, the first two inference mechanisms from D6.1 (like-based and topic-based) will be treated as special cases of a general user attribute classification approach. In addition, two approaches will be examined for improving classification accuracy: feature fusion and multi-target classification.

The features used in the experimental study include the following:

- **likes**: Ids of Facebook likes that appear in the trails of at least two users. There are 3,622 such likes.
- **likesCats**: Facebook categories that user likes belong to. These are directly provided by Facebook and correspond to general categories, such as "Community" or "Music". The likesCats vector for a user is built by counting the number of categories of all the

likes made by the user and then doing L2 normalization. Only categories appearing in the likes of at least two users were retained (191 dimensions).

- **likesTerms**: The terms appearing in the description, title and about sections of the likes made by users, which is equivalent to a classic Bag-of-Words model. We kept only terms appearing in at least two users' trails, which resulted in a vocabulary size of 62,547 terms (after stop-word removal that was carried out using three lists of stop words for the three main languages that appear in the collected content: English, Dutch, Swedish[4] language-specific list of words).
- **msgTerms**: The terms appearing in the posts of users, again using a Bag-of-Words model and the same pre-processing as for likesTerms resulting in 24,990 terms.
- **LDA-t**: The topics extracted from both the textual content posted by the user and the description, title and about sections of the likes made by users. Topics are extracted using Latent Dirichlet Allocation (Blei et al., 2003) and for each user, a topic distribution is computed. Different setups involving different numbers of features are examined (t = 20, 30, 50 and 100 topics).
- **visual**: The visual concepts extracted from the images posted by users. Visual concepts are extracted using the Caffe-based (Jia, 2013) approach that was presented in D5.2. For each image the 12 most dominant concepts are kept and are aggregated in different ways:
    - **visual-bin**: Build a binary vector representing the presence or absence of concepts in the images of a user.
    - **visual-freq**: Build a numeric vector that keeps the frequency of appearance of each concept in the set of images of the user.
    - **visual-conf**: For each concept-image pair, the confidence that the concept is represented in this image is available. The third option is to sum these confidence scores over all images for each concept.

In the following experiments, we considered 97 questions from the questionnaire. These questions correspond to nine of the 10 disclosure dimensions (listed in Annex I). It should be noted that, some of the questions are indirectly mapped to user attributes, e.g. the questions related to personality traits, and location is the only dimension that is not considered here since it is inferred with a specialized method that is developed within WP5. For an early version of the questionnaire please see D4.2. As mentioned, data was collected for 170 distinct people. Training and testing was done using repeated random sub-sampling validation. In this procedure, the data is randomly split *n* times into training and validation sets. For each split, a model is fit to the training set and its prediction accuracy is assessed on the validation set. The final performance is calculated as the average of the *n* performance assessments. For this study, 66% percent of the data were used for training and the process was repeated 10 times. We selected weighted (by class size) Area Under ROC (wAUC) (Li, 2010) as evaluation measure. The reason for preferring wAUC over the more standard measure of classification accuracy was the fact that for many of the questions (user attributes), the distribution of responses was highly imbalanced. For instance, Figure 5 shows the distribution of responses to the question *"How would you say your health is"*. Clearly there are far fewer people that respond that their health is poor than people that respond that their health is very good. In such cases, using classification accuracy for

[4] http://www.ranks.nl/stopwords

quantifying a classifier's performance is known to be problematic because classifiers that tend to frequently predict the majority class, receive a misleadingly high score while not being useful in practice; wAUC is a much better alternative in such cases.

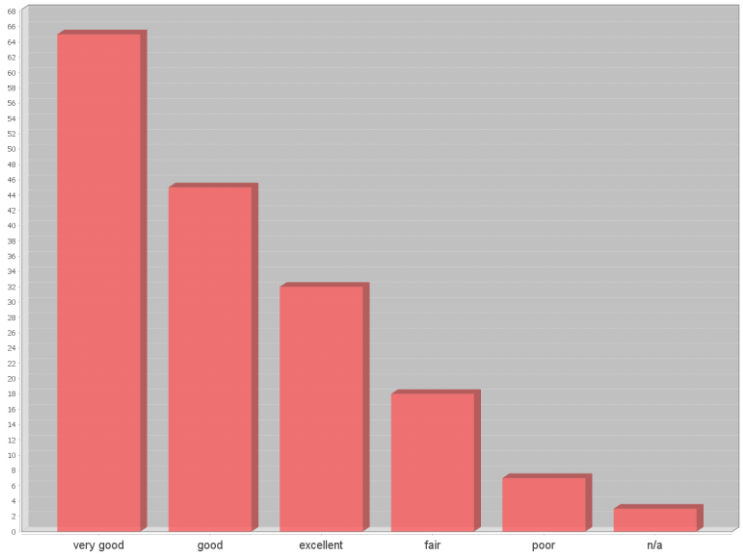


Figure 5. Distribution of responses to the question "How would you say your health is?"

3.2. Results

In the first set of experiments, a number of different classifiers are evaluated. In particular the following classifiers are examined:

1. **zeror:** This classifier always predicts the majority class in the training set. It effectively ignores the input and is used as baseline.
2. **knn:** The k-nearest neighbours classifier (k=10 was used).
3. **tree:** A decision tree classifier.
4. **nb:** The Naïve Bayes classifier.
5. **adaboost:** The Adaboost M1 meta-classifier boosting mechanism with a decision stump (a one-level decision tree) as the base classifier (Freund, 1996).
6. **rf:** The Random Forest classifier using 100 random trees (the use of 10 trees was also tested but resulted in worse results in all cases).
7. **logistic:** A very efficient implementation of L2-regularized Logistic regression from LibLinear (Fan et al., 2008) with probabilistic estimates and tuning of the regularization parameter (values tested: 0.1, 1, 10).

In this experiment, since the focus is on comparing the performance of different classifiers, only a subset of the 97 attributes were studied[5]: eight questions (user attributes) were selected (please see Table 1) and for each of those, all features and classification algorithms listed above were evaluated. Results are shown in Figure 6. The best wAUC by considering all types of features is shown for each attribute-classifier pair; note that results are grouped by attribute.

[5] In subsequent experiments, all 97 features were considered.

BMI: • Underweight • Normal • Overweight • Obese	Income: • 0-999 • 1000-1499 • 1500-1999 • 2000-2499 • 2500-2999 • 3000-3499 • 3500-3999 • 4000-4999 • 5000 or more
Health status: • Very good • Good • Excellent • Fair • Poor	Use of cannabis: • Yes • No
Smoking behaviour: • Never smoked • Used to smoke sometimes but not now • Used to smoke regularly but not now • Smoke, but not as many as one per day • Smoke, between 1 and 10 cigarettes per day • Smoke, more than 10 cigarettes per day	Employment status: • Employed • Unemployed • Retired • Other
Drinking behaviour: • I don't drink alcohol • 1 or 2 times in the past year • 3 to 11 times in the past year • Once a month • 2 to 3 times a month • 1 to 2 times a week • 3 to 4 times a week • 5 to 6 times a week • Every day	Sexual orientation: • Heterosexual • Homosexual • Bisexual

Table 1. Eight user attributes and their possible values selected for the first set of experiments

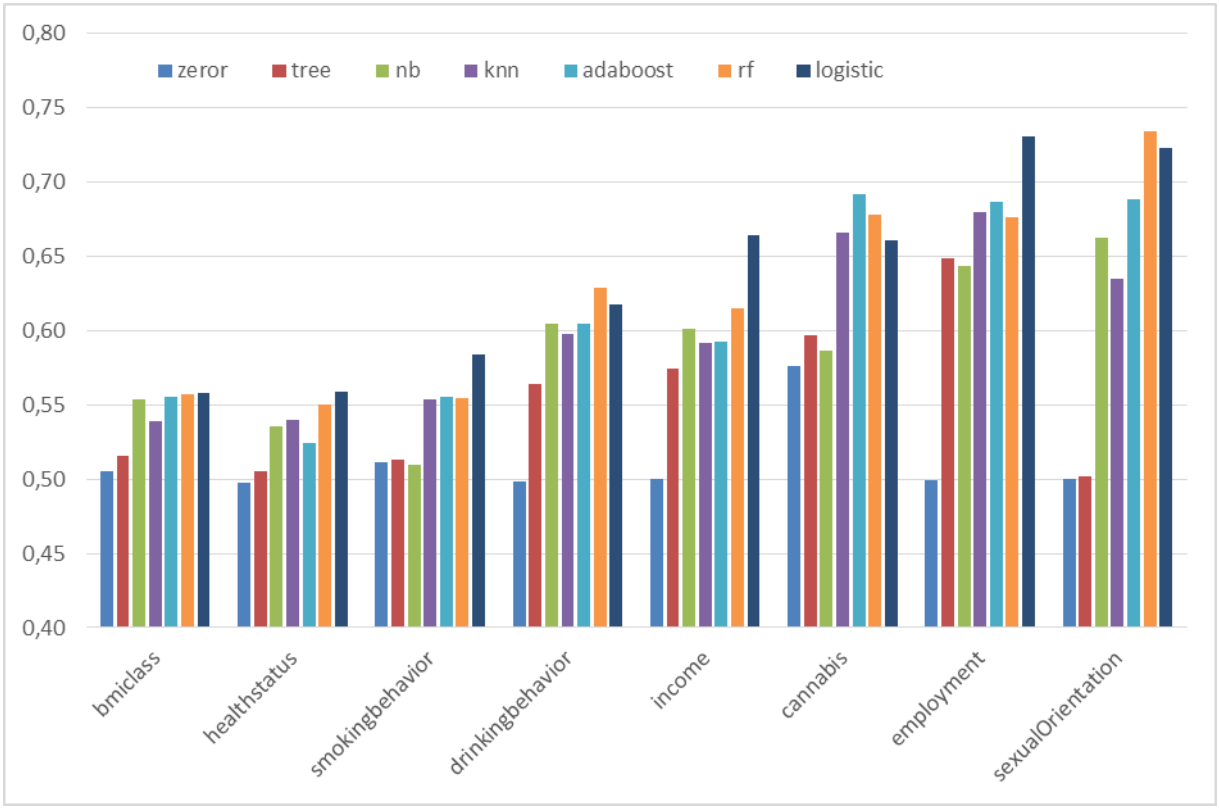


Figure 6. Comparison of performance of six different classifiers on eight attribute prediction tasks.

According to the figure, random forest and logistic regression outperform the rest of the classifiers in seven out of eight tasks. Specifically, logistic regression achieves the best performance in five tasks, random forest in two tasks and adaboost in a single task. Given the good performance of random forest and logistic regression and their better scalability (especially with respect to the number of features) compared to the rest of the classifiers, we opted for using these two classifiers in the rest of the experiments.

In the next experiment, the predictive accuracy of all the aforementioned features was compared. Importantly, all 97 attributes were considered and the performance of the classifiers over all 97 classification tasks was averaged in order to evaluate the contribution of each type of feature. Results are shown in Figure 7.

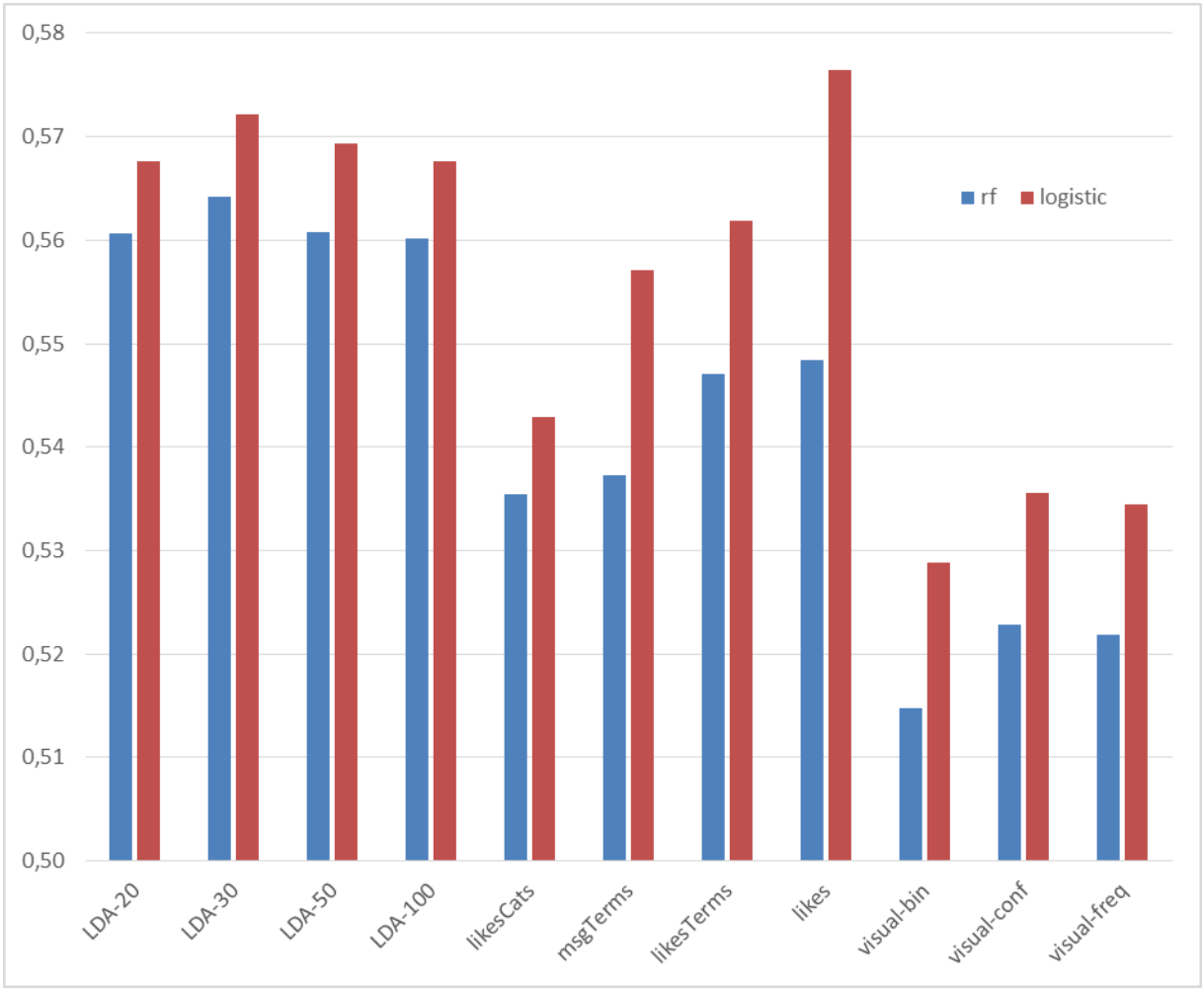


Figure 7. Average wAUC over all 97 classification tasks for each type of feature considered using random forest and logistic regression as classifiers.

We observe that the best performance is obtained using **likes** as features, followed by **LDA** and **likesTerms** features**.** On the other hand, features based on visual concepts obtain lower performance scores, indicating that it is difficult to predict user attributes using this type of information alone. Among **LDA** features, **LDA-30** has a small edge over other LDA-based features and **visual-conf** obtains the best performance among features based on visual concepts. With respect to the two classifiers, logistic regression is consistently better (on average) than random forest in all cases.

Next, the possibility of increasing performance by combining different features is explored. To this end, early and late fusion were tested. Early fusion is performed by concatenating (and normalizing) the features before feeding them into the classifier, whereas late fusion is performed by averaging the results produced by classifiers built on different types of features. In this set of experiments, we used the logistic regression classifier and evaluated the performance of all possible combinations of two features using early and late fusion. Specifically, all distinct pairs (15) of the following features were evaluated: **likes, likesTerms, likesCats, msgTerms, LDA-30,** and **visual-conf**. Figure *8* shows the average performance obtained by the different early and late fusion schemes, along with the performance of models that are based on single features to facilitate easy comparison.

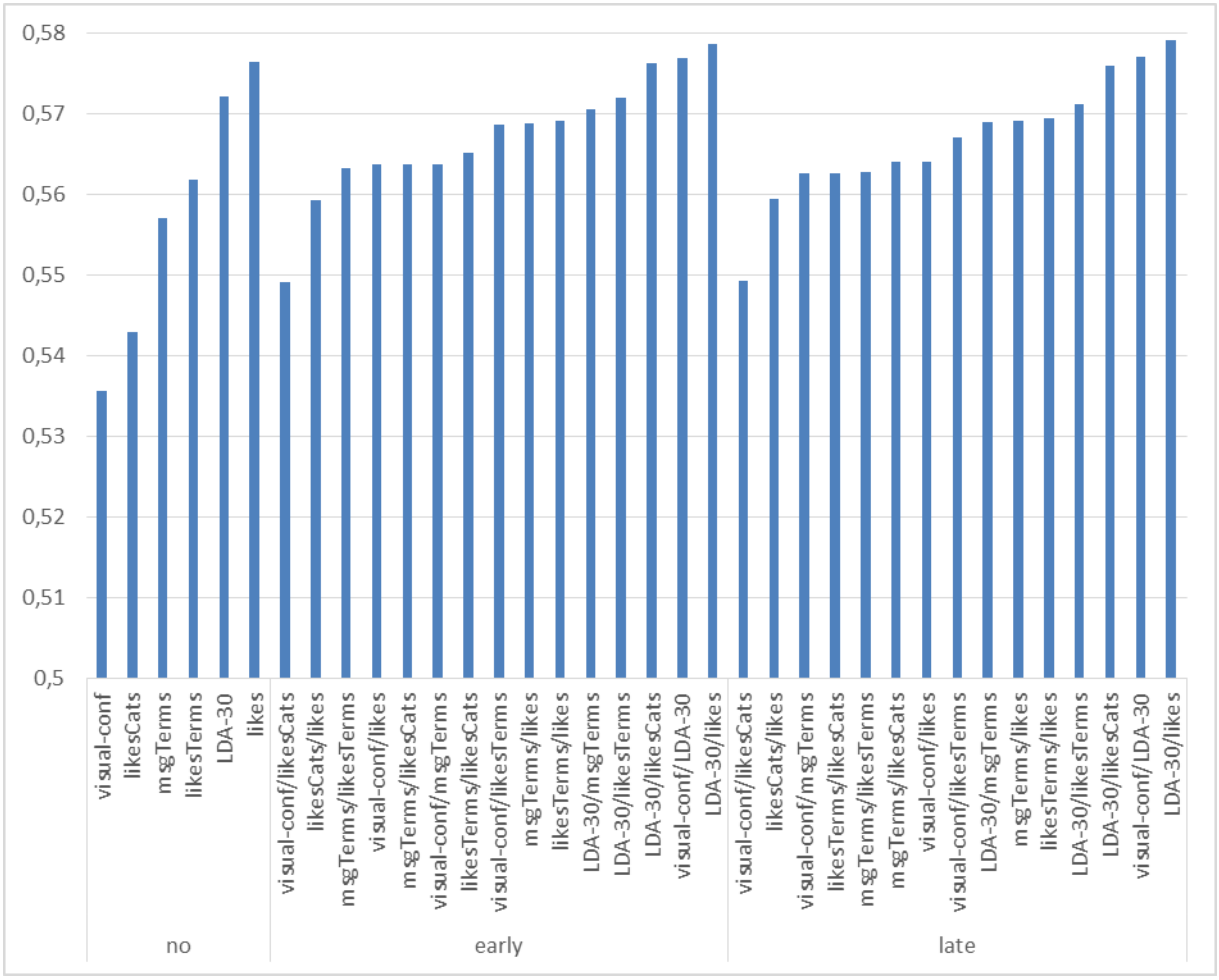


Figure 8. Average wAUC over all 97 classification tasks for single-feature models and models that employ early and late fusion strategies to combine different features.

We see that three early and three late fusion schemes obtain slightly better performance than the performance of the best single-feature model. In particular, the top three results in both early and late fusion come from the same three combinations of features: **LDA-likes, LDA-visual, LDA-likesCats**. Thus, all top-performing combinations include **LDA** features. Another interesting observation is that although **visual** and **likesCats** are the worst performing stand-alone features, their combination with **LDA** features provides better results compared to e.g. the combination of **LDA** features with **msgTerms** features. This is attributed to the fact that **msgTerms** are computed from the same data (terms appearing in a user's likes) as **LDA** features and thus have a lower degree of complementarity with them compared to **likeCats** and especially **visual** features. Comparing early with late fusion, we see that the performance of late and early fusion schemes computed from the same pair of features are very similar. However, late fusion performs slightly better.

It is recognized that different user attributes are very likely to be correlated and therefore predictive performance could increase when considering such correlations. Approaches that deal with the prediction of multiple interrelated variables from a common set of predictive variables can be found in the literature of multi-label classification (Tsoumakas et al., 2010), where all target variables are typically binary. The prediction task that we deal with in USEMP is more general since in addition to binary target variables there are many nominal target variables, i.e. variables with more than two target levels. Thus, many existing multi-

label classification approaches are not applicable, e.g. those that transform the multi-label classification problem into one or more multi-class classification problems where classes correspond to combinations of labels (Read et al., 2008). On the other hand, multi-label classification approaches that build a separate prediction model for each target class can be easily adapted to handle different types of target variables as discussed in (Spyromitros-Xioufis et al., 2012). This category includes the baseline single-target approach (ST) where prediction models for different targets are independent but also approaches that manage to exploit target dependencies by extending the original feature space by treating other prediction targets as additional input variables. A recent approach of this type that has received increased attention from the machine learning community is Classifier Chains (CC). CC constructs a chain of models, where each model involves the prediction of one target and its feature space is augmented by the targets that appear earlier in the chain as additional feature variables. During prediction, where the target values are unknown, CC uses estimates of these values provided by sequentially applying the trained models. This approach was recently extended for multi-target regression in (Spyromitros-Xioufis et al., 2012). Here, we extend it for prediction of a mixture of binary and nominal target variables by employing a multi-class instead of a binary classifier when building models for nominal variables. In addition to CC, we also evaluate an ensemble version of the method called Ensemble of Classifier Chains (ECC) where multiple (instead of one) random chains of classifiers are created and the final prediction for each target comes by averaging the probability distributions generated by each CC model.

We evaluated ST (the single-target approach used so far), CC and ECC (using 10 random chains) on each of the 97 targets using **likes** and **LDA-30** features (the best performing stand-alone features). All three methods can be instantiated with any classification algorithm. For the sake of fair comparison, we instantiated each method with random forest and logistic regression and report, for each target, the best performance obtained using any combination of base classifier and feature. The results on each of the 29 targets related to the consumer profiles of the users are shown in Figure 9 (we do not show results on all 97 targets to improve the readability of the figure).

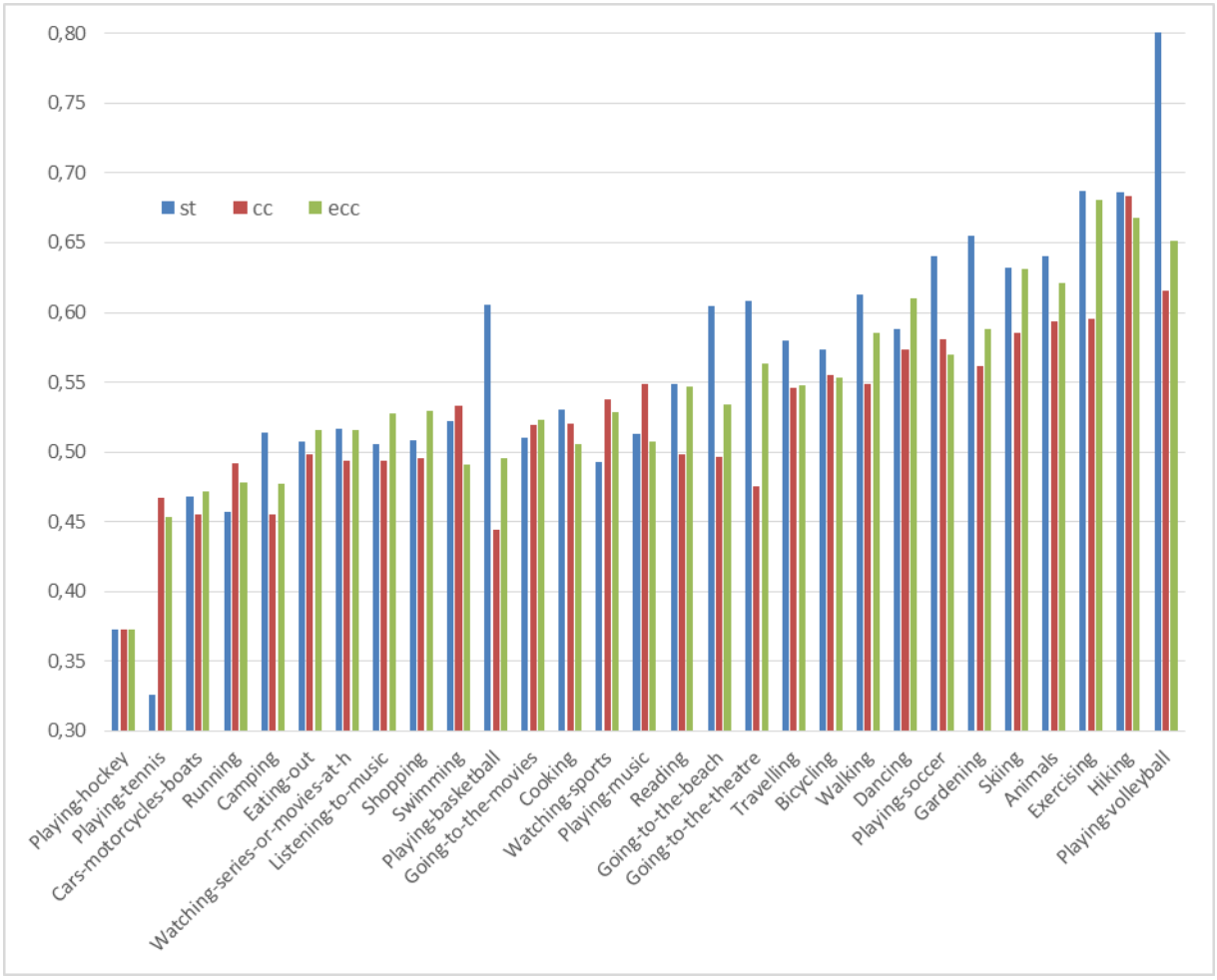


Figure 9. Maximum wAUC per target using ST, CC and ECC.

A conclusion is that CC seems to do a little better than ECC. Nevertheless, on average, there seems to be very little advantage – if any – by applying these two methods on this dataset.

We see that although the performances of CC and ECC are better than the performance of ST in some targets (e.g. 'dancing', 'watching-sports', 'playing-music'), ST performs better on the majority of targets. As expected, ECC performs better than CC on most of the targets. A closer look at the results, indicates that CC models tend to perform better than ST on targets that appear earlier in the chain but the performance starts deteriorating after a certain number of targets has been predicted. This is attributed to the fact that prediction noise is accumulated along the chain. This is a known problem for CC, especially in domains with a large number of targets such as this one. In the future, we would like to evaluate a recent extension (Spyromitros-Xioufis et al., 2012) to deal with this problem of CC. Moreover, we would like to explore the potential of other multi-target prediction approaches on this problem.

The best wAUC for each attribute, using any combination of features, type of classifier and fusion approach are shown in Figure 10. Attributes are grouped according to disclosure dimension. The average best wAUC achieved for all 97 attributes is 0.63. It is important to note here that although predictive performance may not seem very impressive, results are much better than random. Given the rather limited number of data that was available for training and testing, it is reasonable that many classes have been under-represented and it is thus very difficult to learn good representations for them.

Finally, in the early pilot tests, participants were asked whether they thought that information related to each disclosure dimension could be revealed by their OSN data. The questionnaire was designed with the help of the social science partner of the project (see D4.2 and further work in upcoming D4.5). Eventually, these replies were used to rank the dimensions according to the perceptions of users about their disclosure. Table 2 compares this ranking of dimensions to the one according to their actual predictability (resulting from the presented experiments). It is noted that location is missing from the right column because it has not been considered in these experiments.

Ranking	Perceived predictability of disclosure	Actual predictability of dimension
1	Demographics	Demographics
2	Location	Political views (+4)
3	Relationship status and living condition	Sexual orientation
4	Sexual orientation	Employment status and income (+5)
5	Consumer profile	Consumer profile
6	Political views	Relationship status and living condition
7	Personality traits	Religious views (+1)
8	Religious views	Health status (+1)
9	Employment status and income	Personality traits
10	Health status	

Table 2. Ranking of perceptions of users about the disclosure of each dimension and ranking of predictability of each dimension according to the experiments presented in this chapter

Demographics is perceived by pilot test users as the dimension that is more predictable, and indeed it was found through our experimental study that it is the dimension that can be predicted most accurately. On the other hand, **political views, employment status and income appear to be considerably more predictable based on OSN digital trails compared to what test users would expect.**

Pilot test users were also asked about how sensitive they thought information related to each dimension is. The ranking of dimensions according to sensitivity is shown in Table 3. It is interesting to note that the dimension that is perceived as the most easy to predict, that is actually the most predictable – demographics - is considered to be the least sensitive. At the same time, the dimension that is perceived as the least predictable and that is actually one of those that are the hardest to predict – health status – is considered as the most sensitive one. Nevertheless, **some of the dimensions that are perceived as quite sensitive – e.g. employment status and income and political views - can be predicted quite accurately**.

Ranking	Perceived sensitivity of dimension
1	Health factors
2	Employment status and income
3	Personality traits
4	Political views
5	Consumer profile
6	Relationship status and living condition
7	Location
8	Sexual orientation
9	Religious views
10	Demographics

Table 3. Ranking of dimensions according to sensitivity

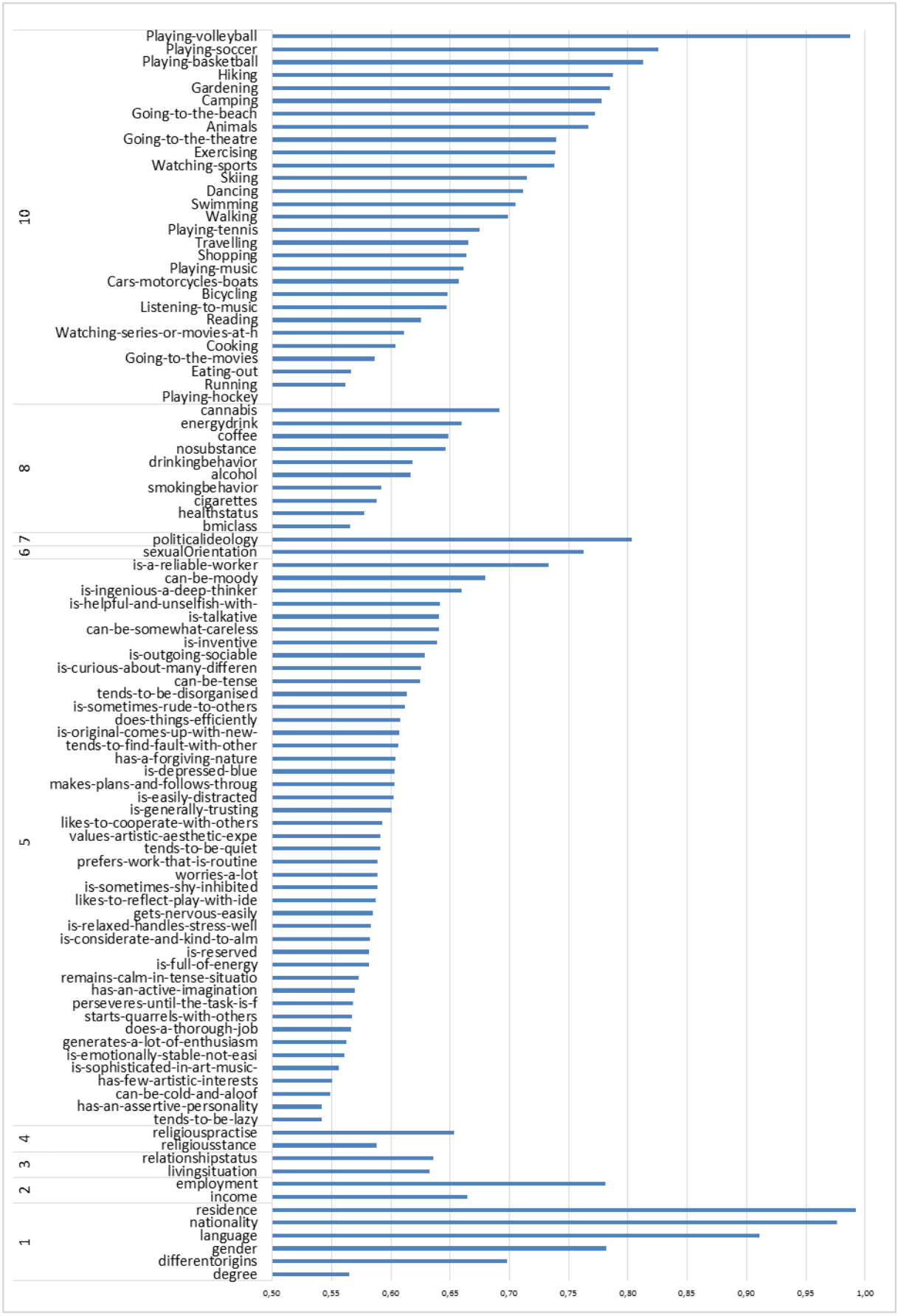


Figure 10. Best wAUC achieved for each attribute using any combination of features, type of classifier and fusion approach.

4. Disclosure Settings Framework

One of the core goals of WP6 is to assist users in their decisions with respect to managing their privacy settings. To achieve this, a user-defined policy was formulated in D6.2 that could represent the sharing preferences of users. This policy is reviewed in this section. A module that materializes this policy was recently implemented and implementation details are presented here. Importantly, a number of limitations imposed on the full realization of the module by the Graph API of Facebook are also discussed. In view of these limitations, a restricted version of the module is also presented. Additionally, in D6.2 a number of automatic and non-automatic methods were identified for assisting users in defining their policy and these ideas were further developed and presented here.

4.1. Overview of disclosure settings framework

In this section, we briefly revisit the basic design of the disclosure settings framework, as originally presented in D6.2 (at that time it was referred to as “privacy settings framework”). The disclosure scoring framework is a user-defined policy that represents users’ information sharing preferences. It is defined as a set of rules, each of which is a triplet consisting of an audience set, content set and access permission.

The audience set defines the set of users on which the rule applies. The content set defines the set of content items posted by the user on which the rule applies. The access permission is simply either “allow” or “deny”. Different ways were identified by which the user could define the audience set and the content set. In particular, the audience set can be defined in the following ways:

- ***Explicitly***: Select a single user or set of users from one’s list of connections.
- By defining a set of criteria based on which users are filtered: either ***demographics criteria***, e.g. attended same school, or more loosely defined criteria expressed as a set of ***keywords***.
- ***Automatically*** *grouping* friends according to the structure of the ego-network of the user, by community detection algorithms, i.e. finding groups of friends that are more densely connected to each other compared to the rest of friends.

The content set can be defined in the following ways:

- **Explicitly**: select a single content item or set of content items.
- Define a specific **privacy dimension, attribute or value**.
- Define criteria based on privacy scores.
- Define a combination of a privacy dimension, attribute or value and a privacy score criterion.
- Define some **keywords** or some temporal criteria.

Once the policy – i.e. a set of rules – is defined by the user, then the system can compute specific sharing decisions for each piece of content posted by the user. In practice, the Graph API allows limited access to the user’s privacy settings: a Facebook application can only change the privacy settings of content that has been posted by the same application. That is, the privacy settings of content posted outside DataBait cannot be changed directly by DataBait. Therefore, for content posted outside DataBait, the best thing that the disclosure settings framework can do is to compute sharing decisions but not directly apply them. Instead, the computed sharing decisions can be presented as suggestions to the user.

4.2. Implementation and API limitations

The disclosure settings framework was implemented in the reporting period. In particular, the mechanism that computes the sharing recommendations was implemented and in addition, the format of the user policy was adapted (somewhat simplified) to be more concise and easily accessible. The module will soon be integrated to the actual system, once the appropriate user interface for interacting with it is built. The interface will conduct two main tasks. The first is to create and edit the policy; at the moment, the only way to edit the user policy is to directly edit the JSON file where it is stored. The second is to communicate the results of the policy to the user. Work is currently being carried out to this end and it is expected that in the following months the module will be fully integrated to the system.

As mentioned, the policy is expressed in JSON format. An original version of the format was defined in D6.2. This has been slightly updated: it has been somewhat simplified in order to produce a more compact and clean representation. An instance of a very simple policy can be seen in Table 4.

```
{
  "policy":{
    "rules":[
     {  "audience":{"type":"explicit","users":["100","101"]},
        "content":{"type":"explicit","contentItems":["IM200","IM220","PO101"]},
        "permission":"allow" },
     {  "audience":{"type":"keyword","keywords":["home"]},
        "content":{"type":"explicit","contentItems":["IM100","IM101"]},
        "permission":"deny" },
     {  "audience":{"type":"demographic","locationCriterion":"Nijmegen","ageCriterion":"20",
                    "genderCriterion":"male","interestsCriterion":["volleyball","reading"]},
         "content":{"type":"explicit","contentItems":["IM202","IM220","PO101"]},
         "permission":"allow" },
     {  "audience":{"type":"group","groupID":"3"},
        "content":{"type":"explicit","contentItems":["IM220","PO101"]},
        "permission":"allow" },
     {  "audience":{"type":"explicit","users":["100","101"]},
        "content":{"type":"keyword","keywords":["soccer","football","sport"]},
        "permission":"allow" },
     {  "audience":{"type":"explicit","users":["100","101"]},
         "content":{"type":"scoring","scoringAspect":"demographics"},
         "permission":"deny" }
    ],
    "defaultPolicy":"deny",
    "aggregationPolicy":"denyWins"
    }
}
```

Table 4. Sample policy with six rules in JSON format

This JSON snippet represents a simple policy that consists of six rules, comprising three clauses: “audience”, “content” and “permission”.

An audience clause can be of any of the following four types, roughly reflecting the audience definition methods that were mentioned before:

- **explicit:** In this case the audience clause also has a field named “users”, which is an array containing OSN user ids.
- **keyword:** In this case the clause also contains a field named “keywords”, which is an array containing a number of keywords. In order for a user to match the audience set, his/her profile information should include all the specified keywords.
- **demographic:** In this case the clause also contains at least one of the fields “locationCriterion”, “ageCriterion”, “genderCriterion” and “interestsCriterion”. In order to get a match, a user should match all defined criteria.
- **group:** In this case the clause also contains the field “groupID” that defines a pre-computed group of friends of the user (more details on detection of such groups will be presented later in this chapter).

The content clause can be of any of the following types, reflecting most of the content definition methods that were mentioned before:

- **explicit:** In this case the content clause also has a field named “contentItems”, which is an array pointing to actual content posted by the user.
- **keyword:** In this case the clause also contains a field named “keywords”, which is an array containing a number of keywords. In order for a content item to match the content set, its textual features should include all the specified keywords.
- **scoring:** In this case, the clause also contains a field named “scoringAspect”, which points to the relevant dimension, attribute or value from the disclosure scoring framework. The link of content items is performed using the support field of the values of the framework. That is, all content items in the support under some dimension, attribute or value that have confidence above 0.5 are selected using this mechanism.

It should be noted that the implementation of other combinations of disclosure scoring criteria that were mentioned in the previous list will be completed upon finalization of the integration of the disclosure scoring framework.

The last part of each triplet defines the permission level suggested by the rule and can either take the value “allow” or “deny”. In D6.2 it was also suggested that the policy may also contain rules that indicate actions, such as “delete” or “untag”, rather than just permissions. This idea was dropped because – due to Facebook API limitations – it would not be practical to implement.

Moreover, the policy as expressed in the format presented in Table 4, apart from the set of rules/triplets, also includes two more fields. The first is “defaultPolicy” and defines what happens when none of the rules apply for some particular audience member and piece of content. It can either take the value “deny” or “allow”, with obvious meaning. The second additional field is “aggregationPolicy” and defines what happens when the disclosure settings dictated by different rules do not agree. It can take four values:

- “allowWins”. If there is at least one rule applicable for some particular audience and content that allows access, then access is allowed, regardless of the fact that there may be other applicable rules that would deny access.
- “denyWins”. If there is at least one rule applicable for some particular audience and content that denies access, then access is denied, regardless of the fact that there may be other applicable rules that would allow access.
- “majority”. The permission dictated by the majority of applicable rules is selected.

- "askUser". If there is disagreement between the rules (at least one allow and one deny), then the user is explicitly asked about how to resolve it.

The computation of recommendations based on the policy specification is straightforward. First, it is recognized that the set of possible values that can appear in the audience is the set of friends of the user and the value "public". Also, the set of possible values that appear in the content is the set of content items posted by the user. The computation of the recommendations of the policy involves finding the rules that are applicable for some particular friend of the user (or the "public") and some particular piece of content. A rule is applicable for some friend of the user (or the "public") if he/she is contained in the audience definition and it is applicable for some piece of content if it is contained in the content definition. Checking if these conditions hold is straightforward for all the different types of definition of audience and content. Once the matching rules have been identified, the matching rules with an "allow" permission and those with a "deny" permission are counted and any clashes are resolved using the preference of the user as expressed in the field "aggregationPolicy". This process is shown in algorithmic form in Table 5.

- Input: A policy consisting of a set of rules $R=\{R^1, R^2, \ldots R^K\}$, a default policy DefP and an aggregation policy AggP.
- Input: Content item I access to which is to be decided
- Input: Audience member F to which access to I is to be decided
- Output: A permission access P ("deny" or "allow"). If empty, the user should be consulted.
- P=applyPolicy(R, DefP, AggP, I, F)

```
ApplicableRulesAllow={ }
ApplicableRulesDeny={ }
for each rule R^i in R:
    if (R^i applies to F and I) then
        if(Ri.permission == "allow")
            ApplicableRulesAllow = ApplicableRulesAllow U R^i
        ilse
            ApplicableRulesDeny = ApplicableRulesDeny U R^i
countAllow = count(ApplicableRulesAllow)
countDeny = count(ApplicableRulesDeny)
if((countAllow==0) and (countDeny==0))
    if(DefP="allow")
        return "allow"
else
    if((AggPolicy=="allowWins")&&(countAllow>0))
        return "allow"
    if((AggPolicy=="denyWins")&&(countDeny>0))
        return "allow"
    if((AggPolicy=="majority")&&(countDeny<countAllow))
        return "allow"
    if((AggPolicy=="askUser")&&(countDeny>0) &&(countAllow>0))
        return null
return "deny"
```

Table 5. Applying the disclosure policy for some specific piece of content and some particular audience member

Please note that for the sake of brevity, the actual code that checks if a rule applies to some specific piece of content and some specific OSN user is not presented here. For more details please see the source code of the prototype that accompanies this deliverable. It should also be noted that the policy should be checked for every possible pair of content item posted by the user and audience member (i.e. each friend of the user and the "public"). This could be

demanding depending on the quantity of posted content and the number of friends of the user. Assuming that there are CountI content items posted and that the user has CountF friends, the brute force approach would require running the above code for CountI x CountF times. Nevertheless, it is possible to identify the rules to which each posted content item applies only once, cache the matches and use these results later on. Also, it is possible to do the same with audience members, i.e., identify the rules that are apply to some friend, cache them and then use these results possibly in combination with the caches of the applicability of rules to items.

As previously mentioned, there are limitations as to what can be achieved using the Facebook Graph API. It has already been mentioned that the permissions computed by the disclosure settings framework can be applied on the actual content of the DataBait user only for content that was posted through DataBait. Therefore, the computed permissions for non-DataBait posted content will rather be provided as recommendations, rather than applied on Facebook directly. Also, in practice it turns out that only very limited information can be obtained about the friends of a user by the Facebook Graph API and this is actually limited only to the friends of the user that also use the same Facebook application (i.e. DataBait in this case), which will correspond to only a small part of the set of actual friends of the user.

During early pilot tests, while it was possible to obtain friends of a user, which also had the DataBait application installed, the sparsity of the resulting data, stemming from the very low overlap of people involved in the tests and those that completed the survey, resulted in very little data on friends of users. Therefore, the fine-grained audience selection mechanisms that have been described are currently not applicable in Facebook, as long as the current API limitations hold. The alternative is to treat all friends uniformly and just have two possible audience sets: “public” and “friends”. In fact, the implementation was adapted to this scenario, and this is the version of the module that is planned for integration to DataBait.

A simple package that contains the code and demonstrates the function of the disclosure settings framework is provided with this report and is described in Annex II.

4.3. Automatic privacy settings definition

In D6.2 a number of ways were described for assisting the user in building their sharing policies. These included different mechanisms for supporting the user in defining the audience and content sets of their sharing rules. Some of these mechanisms were automatic – i.e. they did not require the involvement of users – and some non-automatic. In this and the next section, we present our recent work in this field, starting with automatic methods.

4.3.1. Automatic discovery of social circles

One of the most powerful approaches for assisting the user in defining their audience model involves the automatic grouping of their friends. Groups of friends – also referred to as *social circles* – can be used by the current implementation for defining the audience set of a sharing rule. In D6.2 a number of different graph clustering algorithms were examined for grouping the friends of a user. Additionally, an approach that also used the features of friends for clustering – and not only the set of explicit connections between them - was presented. This method worked by predicting, based on user features, if a link should exist between a pair of users in order to fill missing edges. That approach was essentially supervised, as it required an example ground-truth (reference) grouping of friends to learn a model of the “same-group

relationship" based on features. In this period, an improved approach (compared to the one presented in D6.2) that also takes into account the features of users is presented. First, the new approach is completely unsupervised and does not require an example clustering to be provided. Moreover, it can naturally assign friends to more than one circles, and can handle uncertainty since it is a probabilistic approach.

Let us first provide a formal definition of the problem. Given a set of friends of a user $F=\{\boldsymbol{f_1},\boldsymbol{f_2},...\boldsymbol{f_X}\}$, with each friend f_i being represented by a set of Y attributes (e.g., which school they attended, where they reside, what languages they speak, etc.), i.e., $\boldsymbol{f_i}=\{f_i^1, f_i^2,..., f_i^Y\}$, and given the set of explicit OSN relationships between the friends in F, which can be organized in a graph $G=(F,E)$, where F is the set of vertices of G and E is the set of OSN connections between them, the goal is to recommend a set of circles $C=\{C_1, C_2, ..., C_K\}$, each of which contains a number of friends, i.e. $C_i=\{\boldsymbol{f_k}|\boldsymbol{f_k} \in F\}$. The problem is treated as a clustering problem, albeit, it is recognized that a friend of a user may belong at the same time to multiple social circles. For instance, an OSN friend of a user may belong at the same time to the "school friends" social circle as well as to the "relatives" social circle. Therefore, the clustering approach to be adopted should be capable of producing overlapping clusters.

The proposed approach employs a widely used probabilistic model with this property, the Latent Dirichlet Allocation (LDA) (Blei et al., 2003). LDA is typically applied for topic detection in textual corpora and was also used earlier (chapter 3) for extracting topic-based features from the OSN data of users. LDA detects a number of topics, each of which is modelled as a distribution over words and each document is assigned a probability distribution over the detected topics. Effectively, the set of detected topics also defines a soft clustering of documents, in which cluster membership for a document is given by its posterior distribution of topics. In our scenario, the documents represent the friends of the user and the produced topics are interpreted as the social circles.

We take into account both the profile properties of friends and the OSN links between them using an adapted version of LDA. In particular, we represent each friend of the user with a document of which the tokens comprise a) the profile properties of the user, b) the set of identifiers (ids) of neighbouring users, and c) his/her own id. Thus, the vocabulary of the corpus processed by LDA consists of two types of elements, the first is the set of properties and the second consists of user ids. Eventually, the extracted topics will be distributions over both user properties and user ids, representing at the same time both the dominant properties of the users in a circle, as well as the core members of the circle. As will be experimentally demonstrated, the proposed approach achieves results competitive to state-of-the-art (Leskovec and McAuley, 2012), but at much lower computational complexity.

Related methods

Before proceeding with discussing existing automatic methods for social circle detection, we first review a number of studies that examine the criteria that users actually consider when organizing their OSN friends in groups; such criteria would also be useful for automatic methods. One such study (Jones and O'Neill, 2010) identifies six relevant criteria:

1. Existence of cliques, involving groups of friends being highly connected to each other.
2. Tie strength, which in fact may have various aspects: closeness, emotional intensity of relationship, level of trust and frequency of communication.
3. Temporal episodes. Some groups tend to emerge from people that are all present in a significant event.
4. Geographical locations and spatial proximity.

5. Functional roles. Some links in a social network tend to form because they have some particular use or provide some specific service to the user.
6. Organizational boundaries. For instance, people that work in the same company.

Looking at this list one can note that indeed, both connectivity (first criterion) and profile features (reflected in different ways in the rest of the criteria) are actually considered by users when they group their friends.
Interestingly, (Jones and O'Neill, 2010) also argue that the Structural Clustering Algorithm for Networks (SCAN) (Xu et al., 2007), a graph clustering algorithm that was tested in D6.2, is very suitable for capturing most of these factors, and they perform some relevant experimental analysis supporting this claim. Moreover, they identified that users have difficulty grouping particular friends and they found that this was because these friends either had a weak association with any group or they had strong associations with multiple groups. Since SCAN could successfully identify such people as outliers or hubs, it could be useful not only for grouping friends but also for identifying friends to which users should pay more attention when building their privacy policy. SCAN will be used in the experiments that follow.

However, (Jones and O'Neill, 2010) conclude that there are important pitfalls when using completely automated methods for grouping friends. The reason is that different people considered different grouping criteria to varying degrees. Thus, one needs to consider the different prioritization of users, which is hard to achieve in an automated manner.

A further study that examines the mechanisms by which users group their OSN friends is presented in (Kelley et al., 2011). They identify specific types of groups of OSN friends:

1. General friends, with some specific sub-categories: location-based, generic friends, close friends and friends of friends.
2. College friends, either general college friends or college club/group friends.
3. Friends from other education: high school friends and grade school friends.
4. Other categories such as family, work, church and ``don't know'' friends, people that the user hardly knows or has never met in person.

This list indicates that social circles may form due to various factors and contexts, and hence any type of available information has to be taken into account for carrying out the task.

Let us now look at specific methods that are applicable to the problem. Apart from methods that take into account only OSN connectivity (those were examined thoroughly in D6.2), there is a variety of methods that also take into account the properties of users. Instances of such methods are presented in (Ruan et al., 2013; Yang et al., 2013). However, these methods cannot assign items to multiple groups. Other probabilistic methods, including the method presented in (Leskovec and McAuley, 2012), are capable of assigning friends to multiple circles. Those operate by building a generative probabilistic model. Particularly interesting to the work presented here are methods using LDA. These extend LDA by adding a generative process for the network structure. Examples include the Block-LDA method (Balasubramanyan et al., 2010), Relational Topic Models (Chang and Boyd-Graber, 2009) and Topic Link LDA (Liu et al., 2009). Nevertheless, all previous approaches invest resources to explicitly model the network structure formation, which adds significant computational overhead to the process.

Other approaches also take into account the strength of interactions (Dev et al., 2014). Utilizing tie strength is a meaningful option for the task at hand and could be used even for producing very general circles like those suggested by Facebook (close friends,

acquaintances and restricted). That is, tie strength between the user and some friend could be computed and the resulting value could be used to assign the friend in one of these circles. In fact, there are numerous methods that focus on characterizing the tie strength between pairs of friends. (Gilbert and Karahalios, 2009) presents an approach in which tie strength between two users is predicted using a simple linear regression model. Their model takes into account a total of 74 features, including the number of inbox messages exchanged, the number of "social wall" messages exchanged, the number of photos in which both users are present, the number of days since the last communication, various network measures, etc. (Fogues et al., 2014) presents a similar approach but with the number of predictors reduced to 14, leading to a faster system, as the respective OSN API needs to be queried far fewer times. A further approach along the same lines was proposed by (Spiliotopoulos et al., 2014).

Proposed method

One of the key requirements of social circle discovery is the capability to assign friends in more than one circle. Probabilistic methods such as the ones discussed above meet this requirement. However, all these methods dedicate significant modelling capacity to explicitly model the network formation process. The modelling complexity associated with these methods comes at significant computational cost. In fact, as the experiments to be presented later indicate, the newly developed method is able to achieve results comparable to the state-of-the-art (Leskovec and McAuley, 2012) (that is based on explicit models of the network formation process) without resorting to explicit network formation models, and hence at much lower computational complexity.

As already mentioned, the proposed method is based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a very popular topic detection method. LDA defines a generative model of a corpus of documents, in which each document is represented by a distribution over topics, and each topic is represented by a distribution over terms. The LDA model defines the structure of the joint probability distribution of words, topics and parameters but does not specify how inference and learning of the model parameters are performed. In the original paper (Blei et al., 2009) a Variational Bayesian method was utilized, but other approaches have employed different techniques, such as Expectation Propagation (Minka and Lafferty, 2002). There are different quantities that can be inferred from the joint distribution once learning has taken place, but for our purposes the most important ones are the posterior distribution of topics given a document $p(z|w)$ and the distribution of words per topic $p(w|z)$.

With reference to the notation and problem definition that were presented before, one straightforward way to utilize LDA for social circle discovery, taking into account only the friends' profile properties, would be to treat each friend $\boldsymbol{f_i}$ as a document $\boldsymbol{w_i}$, where $\boldsymbol{w_i}$ includes the properties of friend $\boldsymbol{f_i}$. The identified topics would then be interpreted as the social circles. After training the model, the topic distribution for each friend would provide a circle membership assignment.

Let us now consider what we should do if we were to consider only connectivity information. A common idea in graph clustering approaches is that communities tend to have rather many internal connections, i.e. between their members, compared to connections to the rest of the network. This means that nodes that belong to the same community are likely to share a number of common neighbours. In fact, for several communities, it is likely that some nodes will be very central and that most of the other nodes in the community will be connected to

them. These nodes that belong to the same community, just like documents that are related to a given topic will share the vocabulary that is specific to the topic, or in this case, they will share a "vocabulary of common neighbours". Thus, one could apply LDA for social circle detection taking into account only connectivity information by treating again each friend $\boldsymbol{f_i}$ as a document $\boldsymbol{w_i}$, but this time $\boldsymbol{w_i}$ will include the friend's neighbours in the graph, as well as its own id. Formally, we set $\boldsymbol{w_i}=i \cup \{j|(i,j) \in E\}$.

To leverage both sources of information, i.e. profile attributes and friends' connectivity, we combine them using the same framework. Formally, the document $\boldsymbol{w_i}$ for a friend $\boldsymbol{f_i}$ is defined as $\boldsymbol{w_i}= \boldsymbol{f_i} \cup i \cup \{j|(i,j) \in E\}$. The vocabulary for the corpus would then include both user attributes and friends' ids and therefore the posterior distribution of words for a specific social circle would provide both the dominant properties of the users in the community, as well as the dominant users, at the same time.

Considering that combining the two types of features is likely to improve the results obtained from any of the two types of information alone, it is interesting to examine whether other features could be included to further improve performance. One such possibility is to add features obtained from a network-based community detection algorithm that is executed in a preliminary stage. In particular, one could extract the communities of the ego-network, obtain the community ids to which each friend belongs, and then include these ids as additional words in the document for each friend.

To perform social circle discovery, the posterior distributions of topics given a document (or, posterior distributions of circles given a friend) need to be interpreted in order to produce a final assignment of friends to circles. Various approaches are possible, e.g., a possibility would be to examine the entropy of the distribution in order to estimate how concentrated the probability mass is and based on that to decide on the number of circles to which the circle is to be assigned. Nevertheless, we opt for a simpler approach. Each friend is assigned to a circle if the corresponding posterior probability for that circle is above the threshold $1/K$, where K is the number of topics/circles. To select K automatically, we utilize the corrected Akaike Information Criterion (AICc). Models with different values of K are fitted and the one with the lowest AICc is selected.

It should also be noted that, due to the fact that LDA is utilized, the proposed approach can directly take into account only properties of friends that are represented by categorical variables; it is not straightforward to use properties of friends that are represented by numerical or ordinal variables. In principle, such a scenario could be handled either by quantization (and ignoring the order) of the variables, or by extending the LDA model to explicitly take into account such variables. In our experiments, we do not have numerical or ordinal variables, all data are already quantized (and anonymized) by the data providers, so this is not an issue and plain LDA is utilized.

Datasets

We evaluate the proposed method on two datasets that were also used for the evaluation of the state-of-the-art method (Leskovec and McAuley, 2012) on the problem. The first dataset consists of the required profile and network data for 10 Facebook users; that is, the networks around these 10 users and the profile attributes of all the users that appear in these networks. These 10 users also provided a manually produced assignment of their friends to social circles, effectively providing the ground truth for the evaluation of the automatically produced social circles. The second dataset is similar but concerns 973 users from Twitter

and the ground truth has been obtained by fetching lists that the users had already created. The data has been made publicly available by the authors of (Leskovec and McAuley, 2012) after being anonymized; for instance, the actual name of the school that a user has attended is not provided, this is rather replaced by some arbitrary id.

Before reporting on the experimental results, we present some preliminary exploratory results on the data. These results indicate that perfect results cannot be expected based only on either network structure or profile information. We first look at the ground truth for a single Facebook user in the dataset and compute the distribution of profile attributes in each social circle. Then, we compute the Kullback-Leibler divergence between each pair of distributions p_j and p_k, KL(p_j,p_k). Low values of KL(p_j,p_k) indicate that the two distributions are similar.

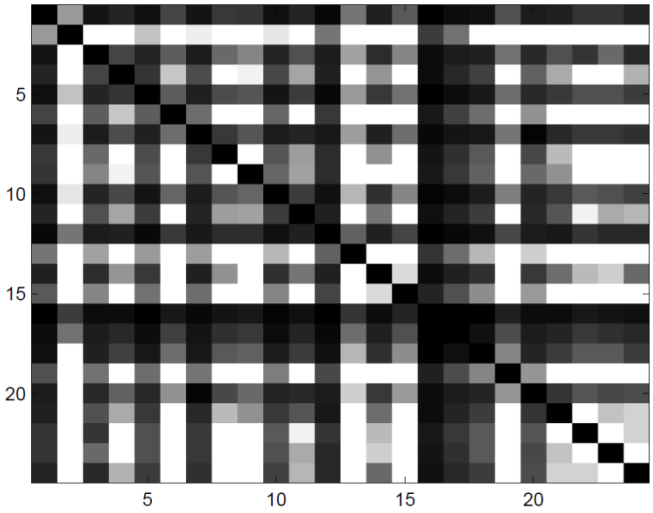


Figure 11. Heat map representing the KL-Divergence of word distributions between the circles of a randomly chosen Facebook user. Dark cells indicate that the KL divergence between the corresponding pair of distributions is small.

Figure 11 depicts in the form of a heat map all pairs of KL divergences for the set of ground truth circles for the randomly selected user. It appears that the distribution for each social circle is quite similar to the distributions of at least some other social circles. Therefore, a method that is based only on the profile attributes of the friends of a user will have to distinguish between distributions that are quite similar to each other. Similar heatmaps are obtained for other users in the dataset as well.

Furthermore, we explore how separated the ground truth circles are, considering only network information. In particular, we utilize the well-known modularity measure (Newman and Girvan, 2004). Modularity ranges between -1 and +1, with positive numbers indicating that the number of edges within groups exceeds the number expected on the basis of chance. In general, positive values will indicate easier to cluster graphs. We compute the contribution to modularity of each social circle of the 10 Facebook users. This gives an indication of how separated each social circle is from the rest. The distribution is shown in Figure 12. As it can be seen, there are only a few social circles with a clearly positive score, meaning that they are very well separated. There are also a few with a clearly negative score, but most of them have a score very close to zero. This indicates that most of the social circles are not very well separated from the rest, considering only network connectivity. In short, these explorative results indicate a) the difficulty of the task and b) the fact that neither of the sources of information (profiles and network connectivity) alone is likely to result in perfect results and therefore it may be beneficial to combine them.

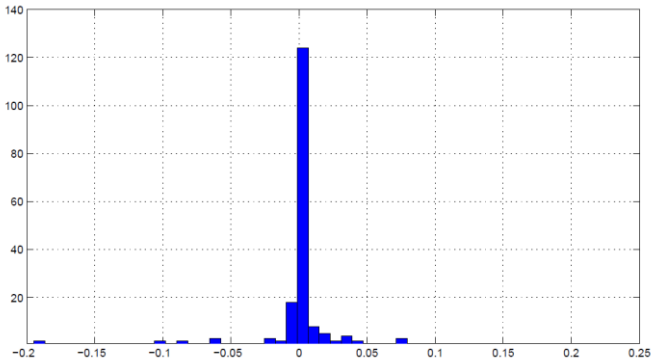


Figure 12. Histogram representing the distribution of modularity contributions for individual circles in the Facebook dataset.

Experimental setup

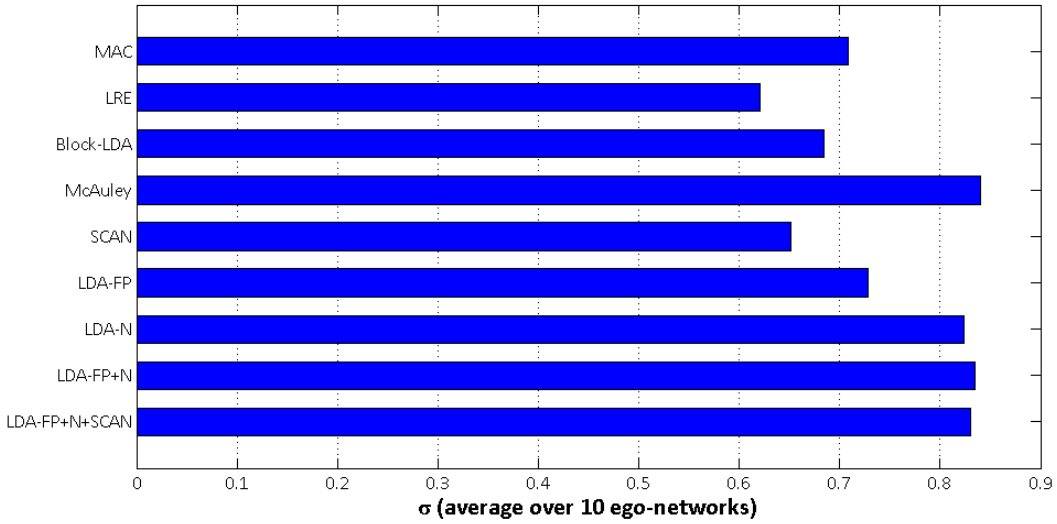
To be able to directly compare our results to those reported in (Leskovec and McAuley, 2012), the same evaluation metric, Balanced Error Rate (BER) is used from the evaluation script that was provided by the authors of (Leskovec and McAuley, 2012). We evaluate and compare five state-of-the-art approaches, with four instances of the proposed approach:

- Multi-Assignment Clustering (MAC) proposed by (Streich et al., 2012), which operates on the vectors of profile attributes (i.e. does not use connectivity information) and assigns each corresponding node to multiple clusters.
- Low-Rank Embedding (LRE) (Yoshida, 2010), in which both the node attributes and the connections between them are projected into a feature space, where classical clustering techniques are applicable.
- Block-LDA (Balasubramanyan et al., 2009), a generative model-based technique that jointly considers profile attributes of users and their connections.
- SCAN community detection (Xu et al., 2007). SCAN depends on two parameters, μ and ε that control the minimum size and density of communities. We report the best results obtained by tuning the two parameters (values examined for μ ranged from 2 to 5 and from 0.3 to 0.8 with a step of 0.1 for ε).
- Social circle discovery method by (Leskovec and McAulley, 2012), which takes into account both the users' profile attributes and their connections.
- LDA using only the set of friends' profile attributes to construct the documents. We will refer to this method as LDA-FP.
- LDA using only the network structure to construct the documents: each document consists of the ids of the corresponding user's neighbours and the user's own id. We will refer to this method as LDA-N.
- LDA using both the set of friends' profile features and the network structure. We will refer to this method as LDA-FP+N.
- LDA using the set of friends' profile features, the network structure and the ids of communities assigned to friends by the SCAN algorithm. We will refer to this method as LDA-FP+N+SCAN.

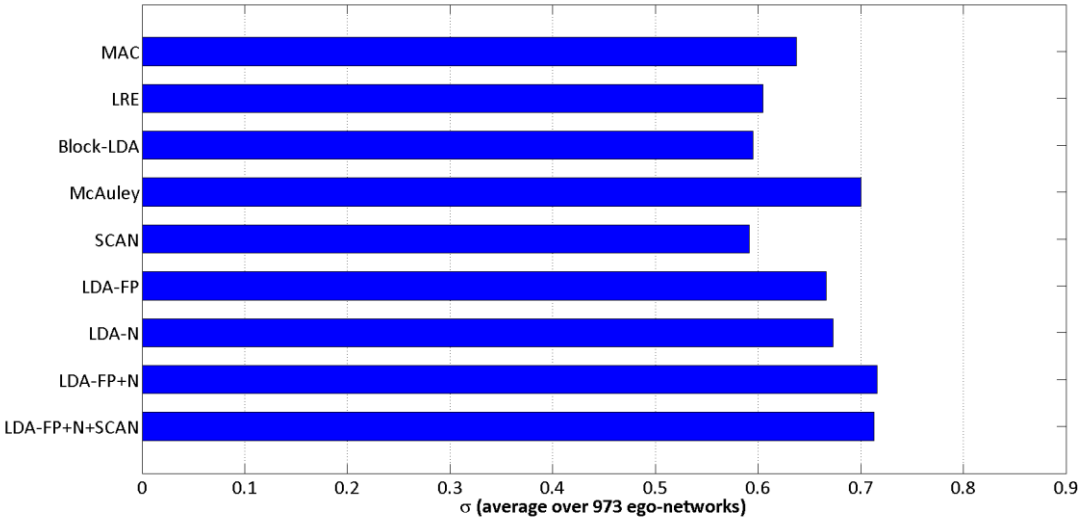
The results of the first three methods (Streich et al., 2012) (Yoshida, 2010) (Balasubramanyan et al., 2009), as well as of the method by (Leskovec and McAuley, 2012) have been obtained from the report of the latter (Leskovec and McAuley, 2012).

Results

Figure 13 shows the performance of the tested methods on the Facebook (top) and the Twitter (bottom) ego-networks. Higher values correspond to better social circle accuracy.



(Facebook)



(Twitter)

Figure 13. Results on the Facebook (top) and Twitter (bottom) datasets

The first observation is that for both OSNs the methods combining profile and network information are competitive to the method of (Leskovec and McAuley, 2012). More specifically, for the Facebook ego-networks, the latter reports a score of 0.84, while LDA-FP+N achieves an average score of 0.8343 (standard deviation is 0.0590) and LDA-FP+N+SCAN achieves a score of 0.8406 (st.dev. 0.0637). For the Twitter ego-networks, (Leskovec and McAuley, 2012) report a BER score of 0.70, while LDA-FP+N achieves a score of 0.7161 (st.dev. 0.1049) and LDA-FP+N+SCAN 0.7127 (st.dev. 0.1171). That is, LDA-FP+N and LDA-FP+N+SCAN slightly outperform the state-of-the-art method on the Twitter ego-networks. Also, based on a t-test, these two methods perform significantly better than the other competing approaches with confidence higher than 99%. It should also be noted that the performance of all methods on the Twitter ego-networks is lower compared to

the performance achieved on the Facebook ego-networks. According to (Leskovec and McAuley, 2012), this is due to the fact that many Twitter circles have not been maintained since they were created (hence they may not reflect reality very accurately), whereas the Facebook circles were created on demand for the purposes of the study.

In addition, LDA-FP, which utilizes only profile information, and SCAN and LDA-N, which take into account only network information, are outperformed by the methods that take into account both (LDA-FP+N and LDA-FP+N+SCAN). Nevertheless, for the Facebook dataset, LDA-N is close to the performance of LDA-FP+N and LDA-FP+N+SCAN; instead, on the Twitter ego-networks, LDA-N achieves somewhat lower performance.

It is important to note that the selection of the number of circles is performed independently for each user and therefore we get a range of values for *K*. For instance, for the 10 Facebook users, the number of produced circles ranges from 2 to 26, whereas for the 973 Twitter users it ranges from 1 to 13. Importantly, when fixing *K* for all users, the results were significantly poorer regardless of the selected value of *K*; e.g. the best performance achieved with any of the LDA-based methods was not higher than 0.72 and 0.61 on the Facebook and Twitter dataset respectively.

Table 6 presents the top words for a random sample of topics produced by the LDA-FP+N approach. Since the dataset is anonymized, profile properties appear with a category name and a numerical value identifier, whereas friends are identified by a single number. One may note that there are circles that are primarily characterized by profile attributes, others that are primarily characterized by user ids (i.e. connectivity) and others by both. More details about this approach can be found in the recent publication (Petkos et al., 2015b).

Table 6. Top words for a random sample of produced social circles using the LDA-FP+N method.

Circle	Top words
1	locale_127 gender_78 educ_type_53 educ_type_55 work_end_157 gender_77
2	67 122 200 21 277 188
3	locale_127 gender_78 educ_type_54 gender_77 school_id_52 educ_year_66
4	53 242 346 249 80 94
5	gender_78 332 locale_127 work_employer_50 339 324
6	educ_type_53 gender_78 educ_concentration_14 educ_type_55 school_50 locale_127

4.4. Non-automatic privacy settings definition

We now look at non-automatic mechanisms for assisting users in defining their disclosure policy. In D6.2 different ways by which the user can be assisted to define the content set and the audience set of rules were discussed. The implementation of some of these methods is currently being considered. In addition, in D6.2 it was recognized that users can be assisted in building their policy by providing them with insights about their audience and their influence on it. This work was extended in two different directions: a) understanding and monitoring the influence that a user has on his/her friends, and b) creation of a propensity model of content consumption. More particularly, in D6.2, we developed a framework for audience monitoring that was based on a) assisting the user to perceive their audience by providing statistics about the demographics data of their friends and b) a propensity model for content consumption. In the following, we effectively present a continuation of this work.

4.4.1. Monitoring user influence

Monitoring the influence that a user has on his/her friends is useful not only for assisting the user understand his/her influence and relationship to other users, but also for calculating the value of the data the user is sharing. Indeed, monitoring influence in an OSN context is important and there are a number of services that use different approaches to achieve this, as for example Klout[6], SumAll[7] and others. In the following, we present a first set of experiments that took place during this reporting period in order to create three metrics that approximately quantify user influence.

The first approximate metric is corresponds to the number of likes a post gathers. It is essentially an indication of the popularity of the post. While this is a useful indicator of popularity, it fails to capture an important factor, the number of friends that a user has.

Thus, the second approximate metric is calculated as the sum of all likes of a post divided by the number of friends that the posting user has. With this metric we examine the "endorsement" of each post over the audience of the network around the user (i.e. friends).

The third approximate metric is a weighted combination of the previous two, taking into account the *user's importance*. For this experiment, we base the calculation of a user's importance on his/her visibility, i.e. the number of a user's friends divided with the number of all users. The third metric provides an insight not only on the "endorsement" of a post but also whether high-valued users have engaged with it.

To test the above metrics, we used a simple setup with five users and five distinct scenarios. In each scenario, these users are linked in a different manner and different liking patterns are applied. In terms of linking between users, the following variations are considered. In the first, there is a central user connected with all other users, while none of the rest four users are connected to each other (simple 5-node star). In the next variation, three peripheral users are also connected with each other (half-full 5-node star). In the third variation, all users are still connected with the central node as well as all peripheral nodes are connected with all other nodes, creating a clique. Graphs representing the connectivity between the users in these three variations are shown in *Figure* 14.

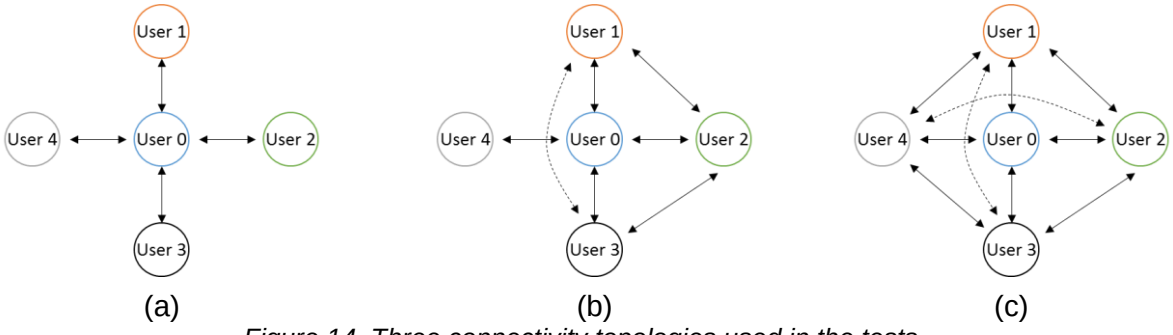


Figure 14. Three connectivity topologies used in the tests.

Eventually, different liking patterns are considered in combination with the above connectivity patterns in order to come up with a total of five scenarios that were eventually examined.

[6] https://klout.com
[7] https://sumall.com

- Case 1: simple 5-node star (*Figure* 14a) where all users like all posts of their connections.
- Case 2: half-full 5-node star (Figure 14b) but the number of likes is still the same as in the previous case.
- Case 3: half-full 5-node star (Figure 14b) where all users like all posts of their connections.
- Case 4: full 5-node star (Figure 14c) but the number of likes is still the same as in Case 3.
- Case 5: clique (Figure 14c) where all users like all posts of their connections

The three aforementioned metrics were calculated for these five cases and the results are visualised in the following figures. *Figure* 15 illustrates the values of the first approximate metric in a radar chart. As expected, User 0, being the central user in all five cases always has maximum influence. As the connections and the likes grow from case 1 to case 5, the influence of Users 1, 2 and 3 also increases, but only reaches the maximum value achieved in case 5. Finally the influence of User 4 changes only for case 5.

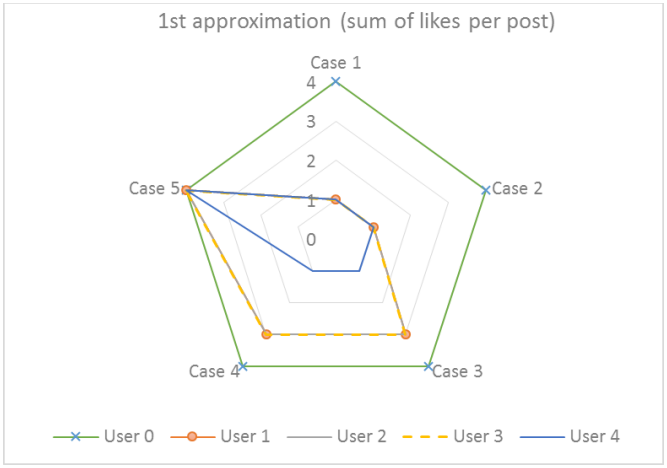


Figure 15. User influence monitoring – first approximation.

The second approximation is depicted in *Figure* 16. In contrast with the previous metric, the influence of User 0 is highly affected by the fact that he/she holds the most central position in the network, and User 4 now has the best influence in all five cases.

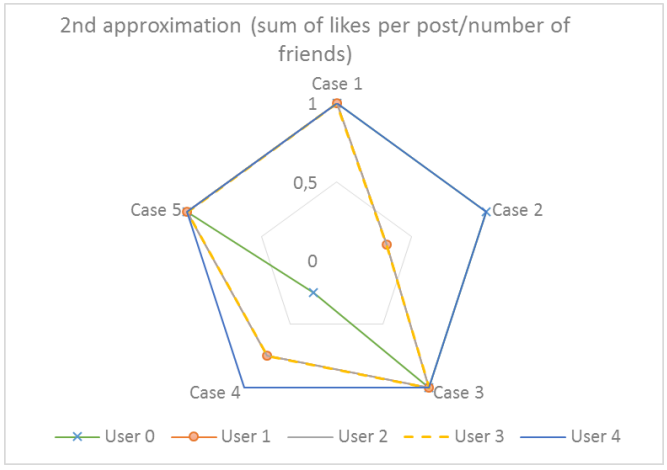


Figure 16. User influence monitoring – second approximation.

Finally, *Figure* 17 presents the third approximation where the user's importance is also taken into account. In this case, the results are similar to the first metric. This seems to provide a better insight not only into the importance of the user, but also its relation to the whole group.

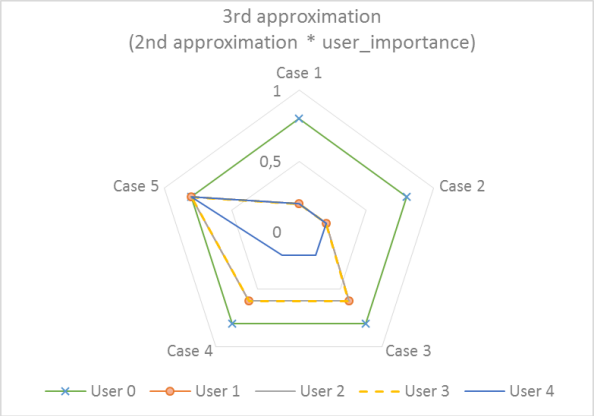
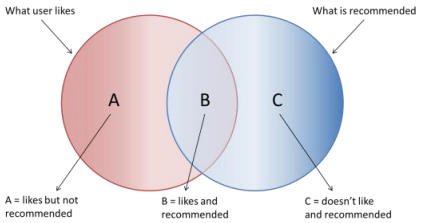


Figure 17. Users' influence monitoring – third approximation.

To conclude, it is clear from the above results that using different influence metrics can lead to significantly different conclusions. It is also important to note that the user's influence is related to the targeting group it is compared to; a User with high influence inside his/her network (e.g. User 4 in Figure 16) might have very limited influence in relation to the rest of the network (e.g. User 4 in Figure 17). In addition, it should be pointed out that other more advanced metrics of influence have been proposed and used in the literature and are likely to be tested by future USEMP work.

4.4.2. User propensity based on content consumption

The second direction of work involves understanding user actions as content consumers. Following the work presented in D6.2 (section 4.4) on audience monitoring, we continued work on generating recommendations for OSN users. As the sample data becomes increasingly bigger, the need to support large sets of data is evident. To this end, a big data back-end was deployed based on Apache Hadoop[8] that supports distributed storage and processing of large data sets on a cluster of computers. On top of the Hadoop system, the Apache Mahout[9] project was used to execute scalable machine learning algorithms for collaborative filtering recommendation algorithms. The results are presented below using precision and recall metrics as presented in detail in D6.2 and illustrated in *Figure* 18.



Precision = % recommendations that are correct

$$P = \frac{B}{B + C}$$

Recall = % correct items that are recommended

$$R = \frac{B}{A + B}$$

Figure 18. Precision and recall metrics.

[8] http://hadoop.apache.org/
[9] http://mahout.apache.org/

The first experiment is related to the precision and recall metrics when using the Tanimoto Coefficient similarity with an item-based recommender. The end goal of the experiment is to study the evolution of the two metrics (precision and recall) when using a daily updated dataset. More specifically, for each of the given days, the data set is updated to include the preferences of users for the previous day and the evaluation of the recommendation is once again executed. *Figure* 19 presents the results of the above experiment in terms of Precision at 2, 5 and 10. The results indicate that variation in the input data is of high importance for the accuracy of recommendations. For both metrics we can see that over the time, where the recommender can be considered more trained and robust, both metrics improve both in terms of performance and of stability, i.e. the variation during the first days slowly smooths out as the sample gets bigger.

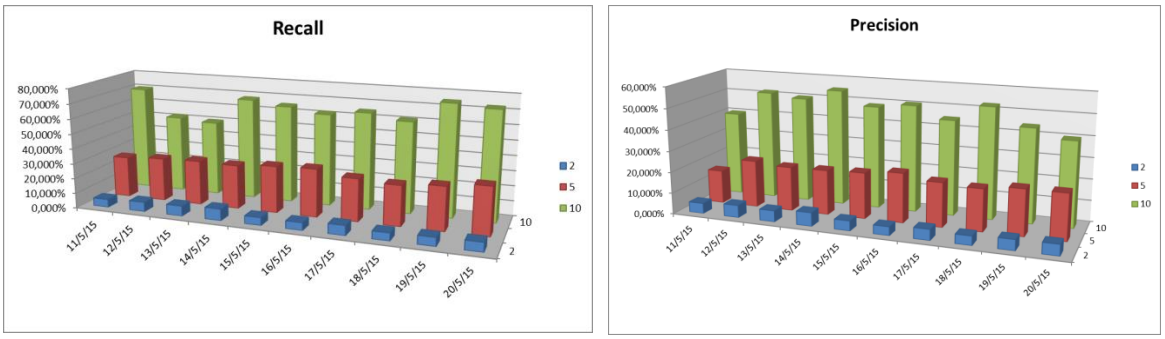


Figure 19. Evolution of recall and precision metrics.

Continuing this line of experiments, in the second example we used a baseline recommender (where the recommendations are chosen randomly) to compare with the performance of the Tanimoto recommender. In this case, we show the performance as the percentage of correct hits (recommendations) comparing the recommendations calculated for the next day given as input all data until that day, compared with the actual users' preference in the next day.

Figure 20 and *Figure* 21 present the performance of baseline and Tanimoto recommender respectively. According to them, the baseline recommendation has low accuracy, which is also highly affected by users' actions. For example, spikes seen during specific days correspond to days where users were more active. The same can be seen also in the case of the Tanimoto recommenders, in which case the performance is considerably improved but still highly affected by users' behaviour.

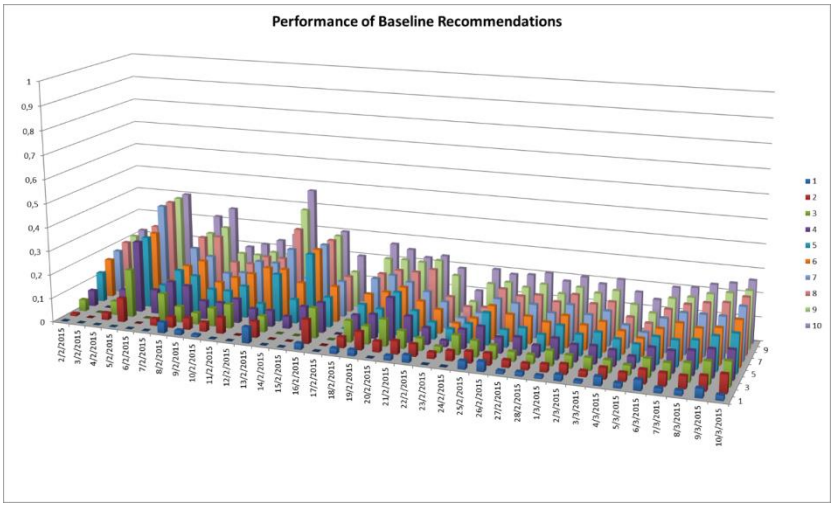


Figure 20. Performance of the baseline recommender.

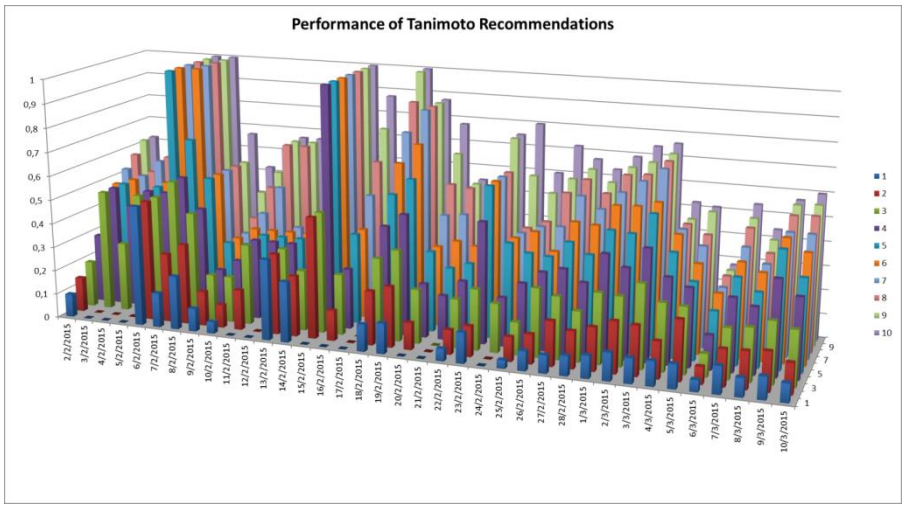


Figure 21. Performance of Tanimoto recommender.

The difference between the two recommenders is presented in *Figure* 22. Overall the Tanimoto recommender is performing considerably better than the random baseline.

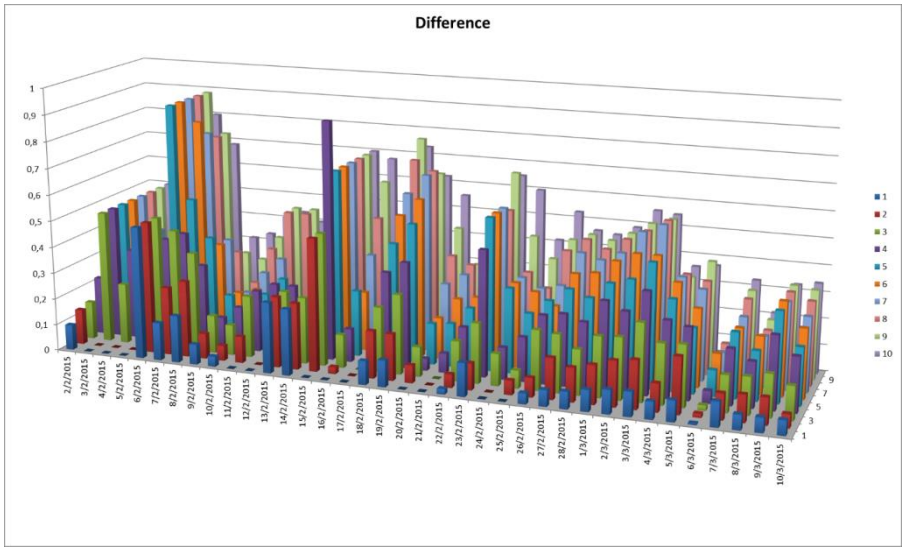


Figure 22. Difference between baseline and Tanimoto recommender.

5. Web Trackers and Do-Not-Track policies

One of the outcomes of the USEMP project is the development of a tool that addresses privacy issues related to a user's Web browsing behaviour. To this end, the DataBait browser plugin allows users to control Web tracking entities that monitor the browsing behaviour of users. In D6.2 we proposed a solution for Do-Not-Track (DNT) policy, which has been implemented by the DataBait Web tracking tools. The rest of this chapter will present the evolution of the DataBait plugin and initial findings in terms of recommending trackers to be blocked by users.

5.1. Enhancements on the DataBait plugin

In deliverable D6.2 we presented the DataBait browser plugin as the "USEMP Web online trackers tools and permissions network". The DataBait plugin is a Web browser extension that allows users to have fine-grained control over third-party trackers when browsing the Web. The plugin provides a visual representation of the trackers and through a user-friendly interface users can select or deselect third-party trackers to be disabled.

During the reporting period, work around the DataBait plugin aim at its full integration with the back-end system. Users of DataBait are also users of the DataBait plugin. When using the plugin, their preferences for blocked trackers are stored in the DataBait back-end, from where they are retrieved when the user re-visits the same site. All stored information and activity of users is made available through the DataBait tool in an aggregated visualisation on which the user can also select or deselect a tracker to be blocked.

It is important to note that the integration with the backend is of high importance in order not only to provide this aggregated form of the blocked trackers, but also in order to provide recommendations for additional trackers to block as will be presented below.

5.2. DNT recommendations, initial evaluation

The end goal of this work is the creation of a recommendation mechanism that will provide users with a set of recommended Web trackers to be blocked. Once again, before integrating any recommendation algorithms to the DataBait plugin, a series of tests and evaluation of different recommendation algorithms needed to take place offline. To proceed with the evaluation, the mahout engine was again used to calculate the precision and recall for the different algorithms as in the previous chapter.

5.2.1. Creating a synthetic data sample

One of the main important factors for any recommendation mechanism is the training set. Using as initial input the data collected by the DataBait plugin from four internal test users, we constructed synthetic mock data to provide the algorithm for training. For the four seed users, there are on average 36 blocked trackers with a deviation of 3. For this experiment, we simulated data for 1000 users. For each new user, a random value of blocked trackers is selected based on a normal distribution with a mean value of 37 and a standard deviation of 5. Each entry of blocked tracker is calculated as a combination of a randomly selected URL and a randomly selected tracker, both coming from the pool of entries of the four initial users.

The recommendation evaluation in mahout was calculated using both an item-based and a user-based recommendation, using a variable portion of the generated data for training (from 10% to 90%). Before presenting the results, we recognize that due to limited real input data,

the generated data may lead to considerably different results. Hence, the following experimentation is meant as a reference benchmark for future experiments, when more real data are introduced in the system by actual users.

5.2.2. Item-based recommendations

Figure 23 presents the evaluation in terms of precision and recall for the item-based recommendation on the mock data of 1000 simulated users using three different similarity metrics: 1) Euclidean Distance, a) precision and b) recall, 2) Tanimoto Coefficient, a) precision and d) recall, and 3) LogLikelihood, e) precision and f) recall. A first conclusion from Figure 23 is that the usage of different percentages of data for training seems to have limited effect on the results in all cases. This implies that the training set of these algorithms can be substantially smaller than the actual dataset. In all three cases we can see that the results are improving for higher values of the “Precision At” parameter. Finally, it is also evident that from the three metrics, both Tanimoto Coefficient and LogLikelihood achieve similar results, while Euclidean similarity results in lower performance.

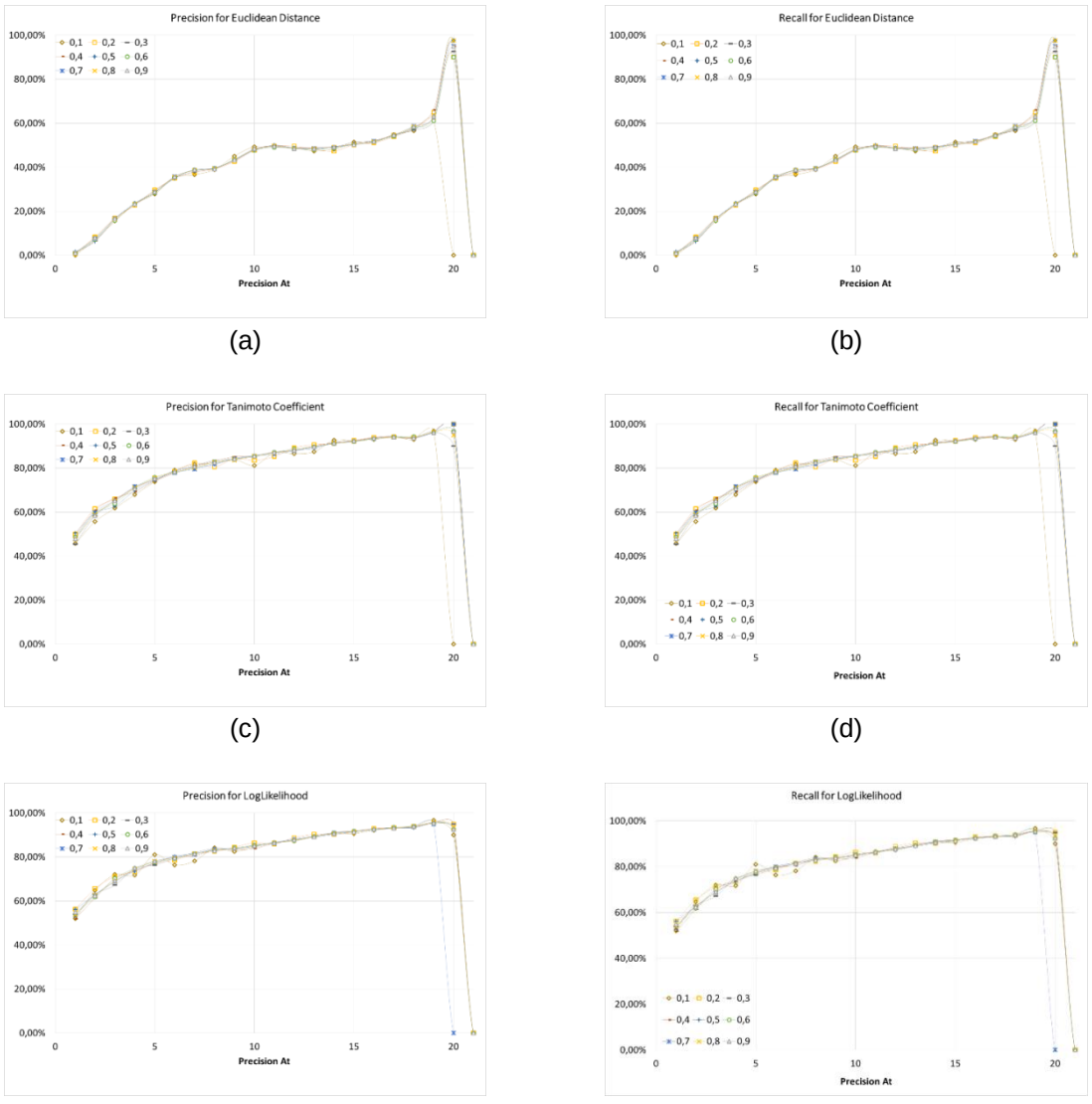


Figure 23. Precision (a,c,e) and recall (b,d,f) for Item-based recommendation on 1K simulated users.

Figure 24 and Figure 25 present in detail what happens in the case where 50% of the sample data is used for training the recommender. Both the Tanimoto Coefficient and LogLikehood

have similar performance, and even for the lowest range of “Precision At” parameter, they exceed 50% in precision and recall, while for the higher ranges they even reach 100%. It should be noted that recommendation accuracy in real data is expected to be lower, and that these unusually high scores are due to the fact that the training set consists of synthetic data. In contrast, Euclidean Distance has much worse performance and in almost all cases, both precision and recall are limited to values <60%.

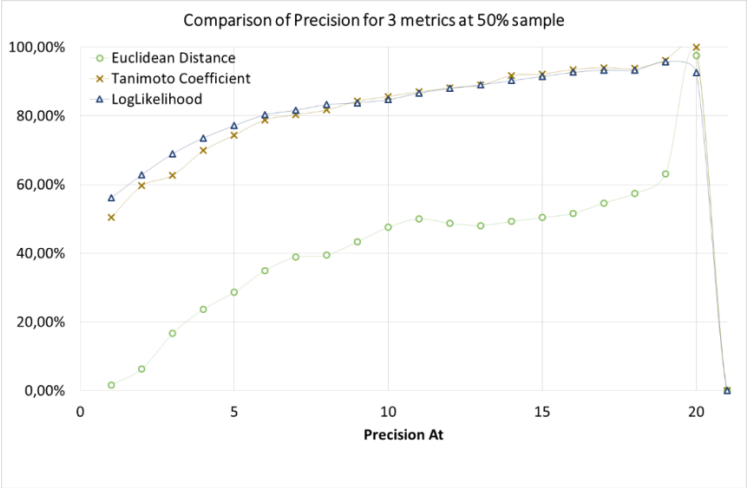


Figure 24. Comparing Precision of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample

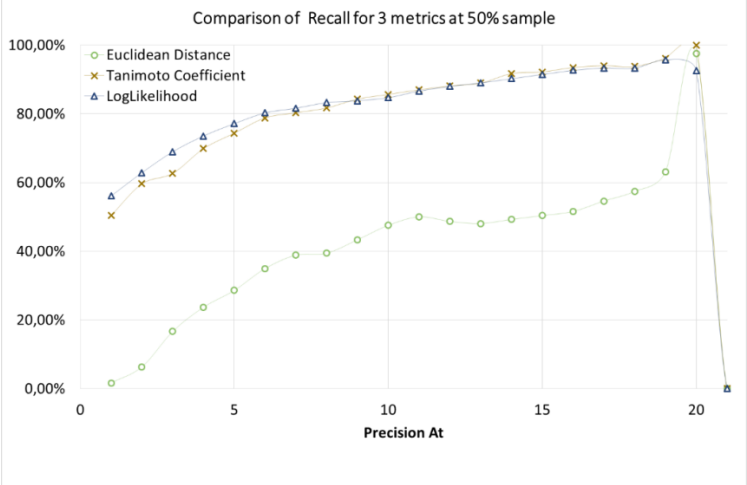
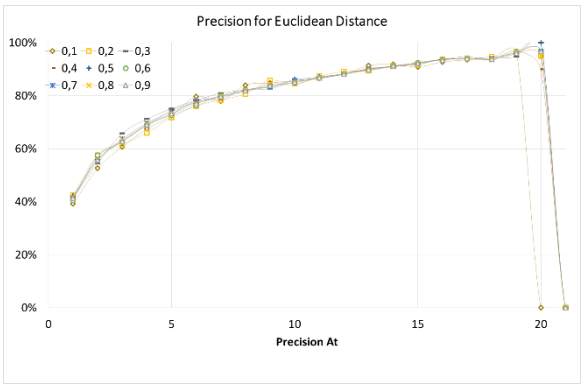


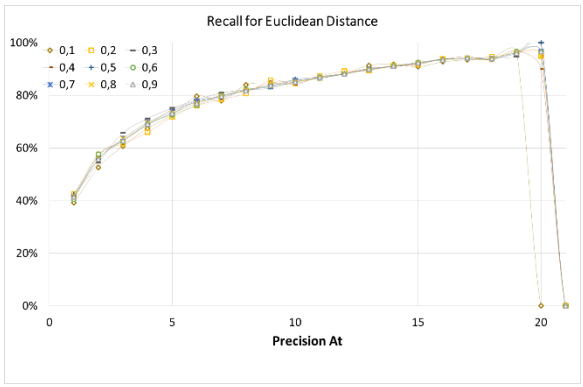
Figure 25. Comparing Recall of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample

5.2.3. User-based recommendations

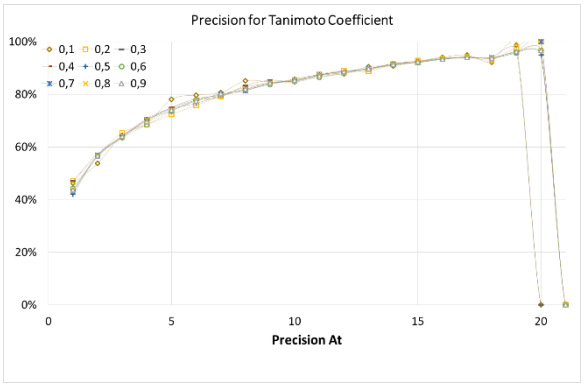
Next, the same experiment was conducted, this time using a User-based recommender. Figure 26 presents the evaluation in terms of precision and recall for the user-based recommendation on the mock data of 1000 simulated users using the same three similarity metrics. Once again, it can be observed that the percentage of data to be used for training has limited effect on the overall performance. Similarly with the item-based recommender, the performance increases for higher values of the “Precision At” parameter. A major difference is that in this case all three similarities lead to similar performance.



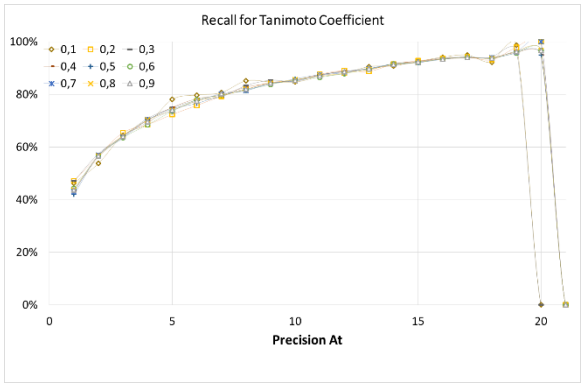
(a)



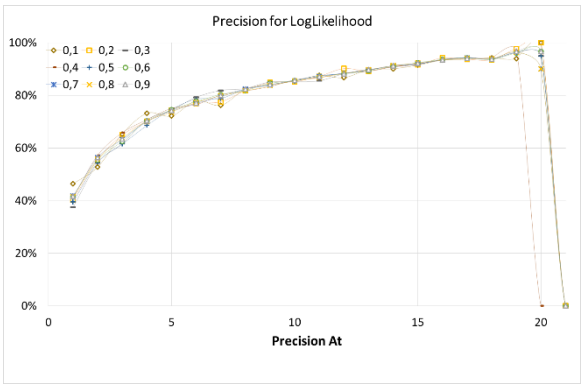
(b)



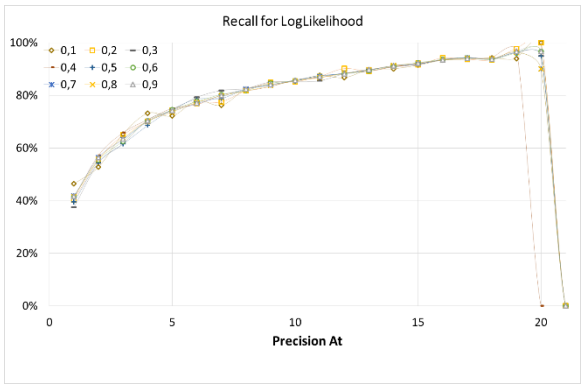
(c)



(d)



(e)



(f)

Figure 26. Precision (a,c,e) and recall (b,d,f) for User-based recommendation on 1K simulated users.

Once again, we used the 50% point, to compare the three similarities and present the precision and recall results in Figure 27 and Figure 28 respectively. All three similarity metrics lead to similar performance. In this case, all three start from precision and recall values around 40% and reach 100% as the “Precision At” parameter increases. Interestingly, the performance of the item-based recommenders is better than the performance of the user-based ones for smaller values of “Precision At”; however, this difference in performance is insignificant for higher values of the parameter.

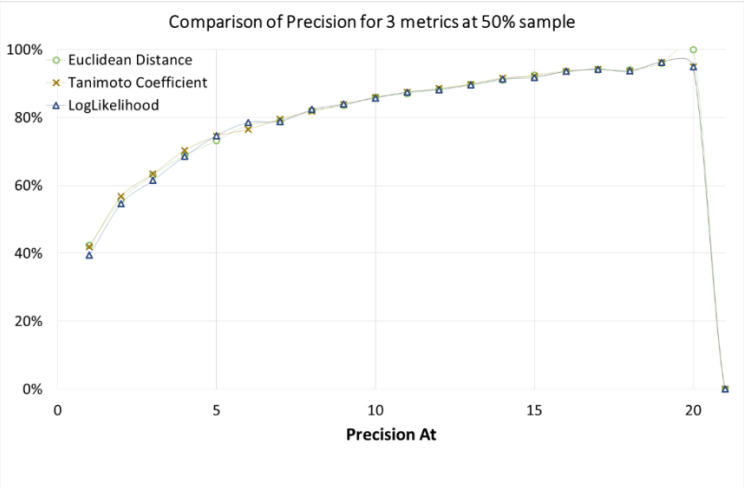


Figure 27. Comparing Precision of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample.

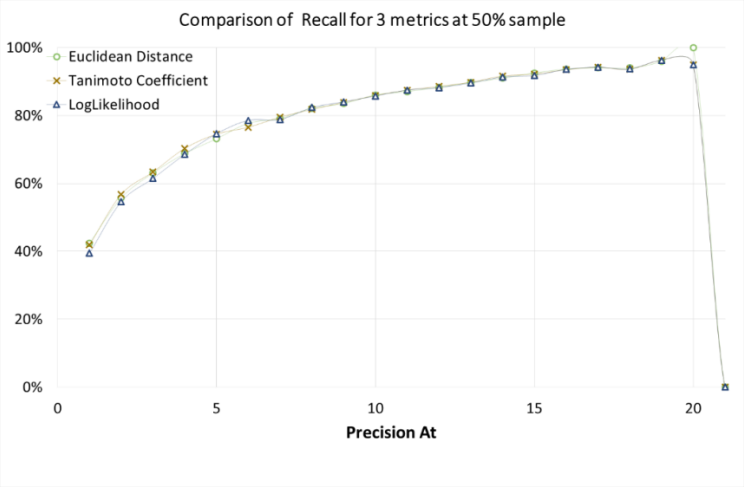


Figure 28. Comparing Recall of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample.

5.3. DNT recommendations, extended evaluation

A second step in the evaluation of the recommendation algorithm is to duplicate the above experiment, this time using a large sample data consisting of 10,000 simulated users. The procedure of creating this sample data is the same as before, with the only parameter changing being the number of users. With more users the recommendations is expected to lead to more reliable conclusions.

5.3.1. Item-based recommendations

Figure 29 presents the results in terms of precision and recall for the item-based recommendation on 10,000 simulated users using the same three similarity metrics that were presented above. In comparison with the results of Figure 23, using Euclidean Distance is limited to a maximum score of 80%. In contrast, the use of Tanimoto Coefficient leads to improved performance especially for the lower values of the Precision At value, while LogLikehood performs similarly.

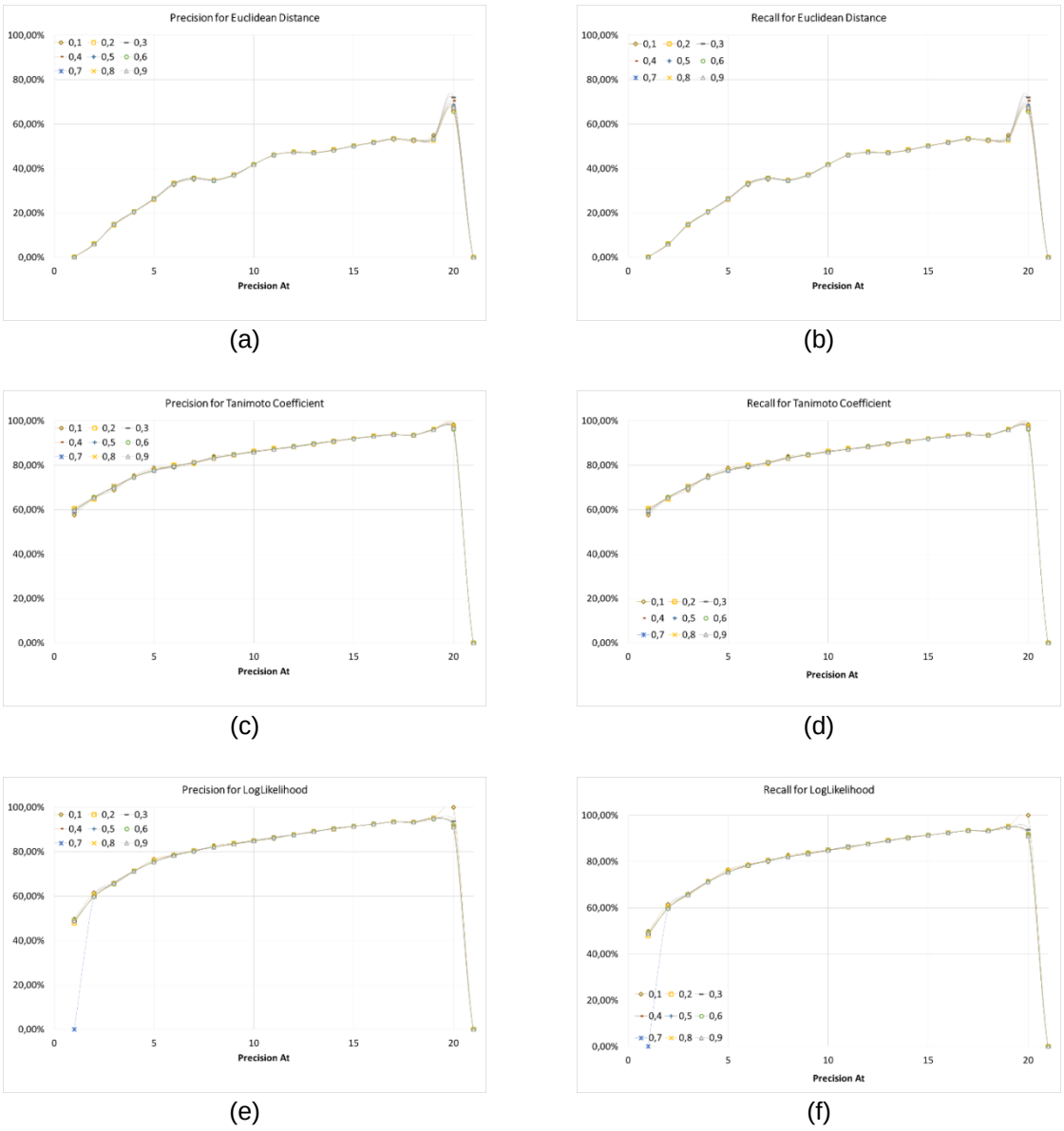


Figure 29. Precision (a,c,e) and recall (b,d,f) for Item-based recommendation on 10K simulated users.

Figure 30 and Figure 31 present in detail what happens only in the case where 50% of data is used for training the recommenders. While LogLikehood and Tanimoto Coefficient had similar performance in the case of 1K users, it is evident that the latter has much better performance in this case.

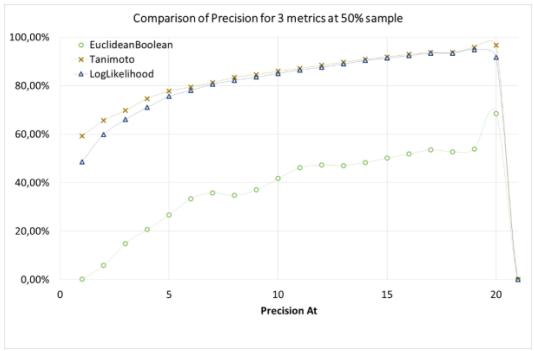


Figure 30. Comparing Precision of Eclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample

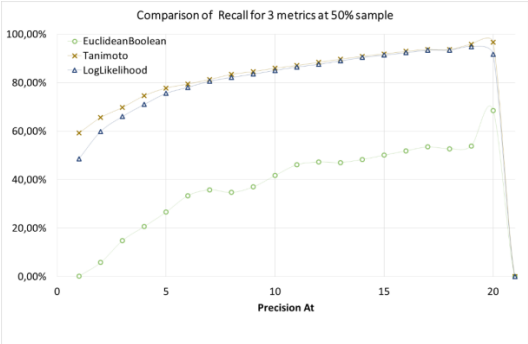
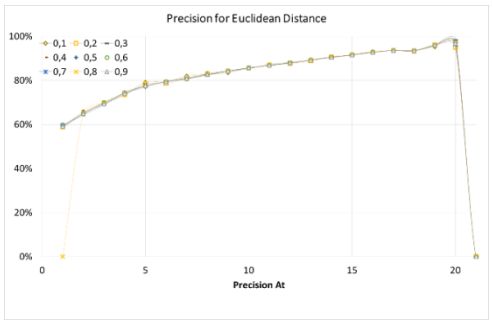


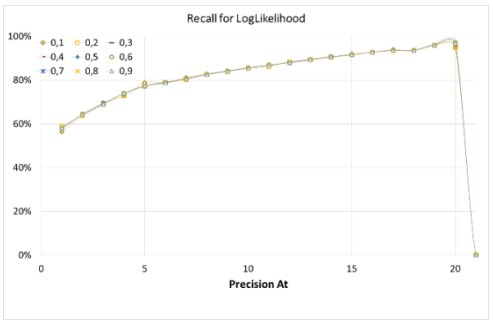
Figure 31. Comparing Recall of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample.

5.3.2. User-based recommendations

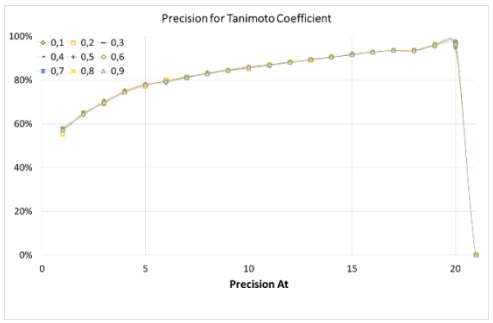
Figure 32 presents the results for the user-based recommendation using the 10,000 simulated users. In contrast with the item-based recommender, the bigger number of users leads to improved performance in all cases.



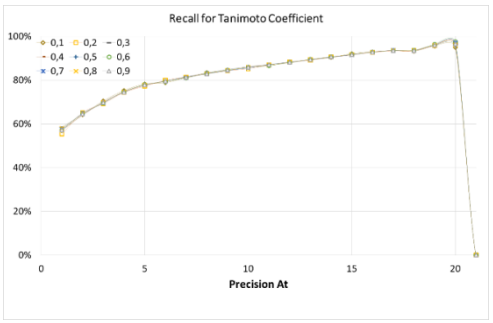
(a)



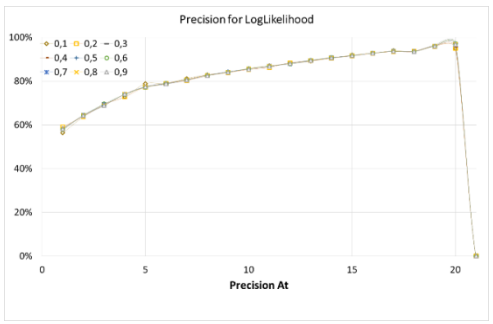
(b)



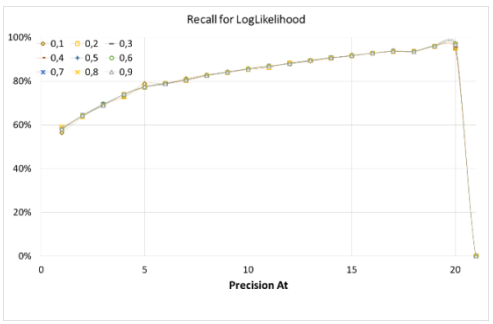
(c)



(d)



(e)



(f)

Figure 32. Precision (a,c,e) and recall (b,d,f) for User-based recommendation on 10K simulated users.

The comparison of performance between the three similarity metrics, when 50% of data is used for training, is shown in Figure 33 and Figure 34. All three metrics lead to similar performance levels, considerably higher in comparison with the experiment involving 1000 simulated users.

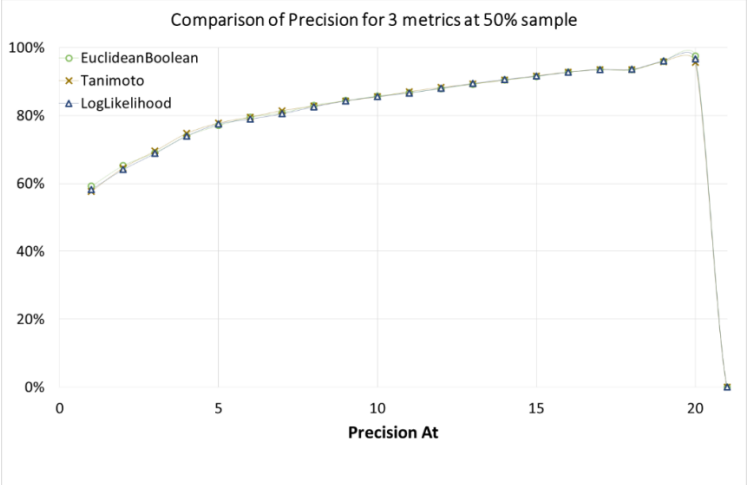


Figure 33. Comparing Precision of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample.

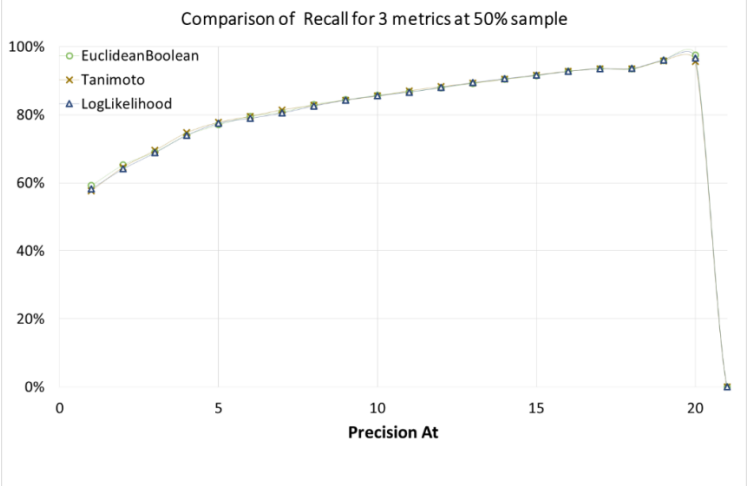


Figure 34. Comparing Recall of Euclidean Distance, Tanimoto Coefficient and LogLikelihood when using 50% of the sample.

6. Conclusions and Next Steps

This deliverable provided an update on the work that was carried out in tasks T6.1 and T6.2 after the submission of the relevant deliverables (D6.1 in M15 and D6.2 in M16) until the end of the second year (M24). Work during this period spanned four main directions, each presented in the previous four chapters. More specifically:

- The disclosure scoring framework, the design of which was originally presented in D6.1 was implemented during this period, and its implementation was made publicly available. The first steps towards integration have already been made and a relevant paper was published (Petkos et al., 2015a).
- Mechanisms for inferring user attributes that were originally presented in D6.1 have been extensively tested with data from the early pilot tests. Various extensions were considered, such as the use of a richer set of extracted features, fusion mechanisms and the use of multi-target classification methods.
- An initial version of the disclosure settings framework that was originally presented in D6.2 was implemented during this period. Additionally, a novel method for grouping the friends of an OSN user into social circles based on both the connectivity between the friends and their characteristics has been developed. A relevant paper was published (Petkos et al., 2015b).
- The Web trackers plugin that aims at supporting users with respect to Web browsing behaviour disclosure was fully implemented and integrated with the DataBait back-end. Additionally, mechanisms for recommending to the user which Web trackers to block have been examined.

Work in the next period will focus on the following activities:

- Full integration of the disclosure scoring framework. This involves four main steps: a) deployment of the implementation in the DataBait servers as a web service, b) link to the privacy database, c) link of the inference modules to the framework (i.e. they need to appropriately invoke the “support” function call) and d) finalization of the development of the visualization and interface.
- Integration of the best inference mechanisms that were identified for each user attribute presented in Chapter 3 to the prototype. Investigation of ways by which the accuracy of inferences can be further improved, e.g. by considering using external datasets for at least some of the attributes.
- Full integration of the disclosure settings framework. This will mainly require effort in the development of the user interface that will be used both for the creation of the policy by the user and for providing recommendations to users. Integration of influence monitoring will also be completed in the coming period.
- Further improvements in the Web trackers plugin, in particular by considering the integration of the recommendations mechanism.

Annex 1 – Privacy Dimensions & Attributes

This Annex provides the updated list of identified attributes for each dimension of the USEMP disclosure scoring framework. This has slightly changed since D6.1 was written, mostly after careful examination of the questionnaire that was provided during the early pilots. The most important change is the addition of two dimensions ("Employment status and income" and "Relationship status and living situation"), the attributes of which were previously under the "Demographics" dimension.

#	Attribute	Description	Example values and range
A.1	Age	Rather than using the absolute number of years, it is typical to use age groups.	The following is a preliminary set of age groups: 6-12, 12-18, 15-25, 25-35, 35-45, 45-55, 55-65, 65-75, older than 75 years
A.2	Gender	The gender of the user	Male, Female
A.3	Nationality	Nationality of the user	French, Belgian, Greek
A.4	Racial origin	The racial background of the user.	Asian, African, Caucasian, Latino/Hispanic, Other
A.5	Ethnicity	The ethnic origin of the user that could relate to a combination of racial background, language and religion.	List of target ethnicities. e.g. Arabic, Easter-European, etc.
A.6	Literacy level	Literacy level will be represented by the highest degree or level of school attended.	None, Nursery school, High school, Bachelor's degree, Master's degree, Ph.D., Other

Table 7. Demographic attributes

#	Name	Description	Example values and range
B.1	Employment status	Whether the person works, is retired, etc.	• Employed • Retired • Pensioner
B.2	Income level	The income level of the person expressed in a set of ranges	• 0-500 euros • 501-1000 euros …

Table 8. Employment status and income

#	Name	Description	Example values and range
C.1	Relationship status	Whether the person is single, married, etc.	• Single • Married • Divorced
C.2	Living situation	Whether the person is living alone, with their parents, etc.	• Living alone • Living with parents

Table 9. Relationship status and living situation

#	Name	Description	Example values and range
D.1	Emotional stability	This describes the overall reaction of a person to negative emotions.	• Sensitive / nervous • Secure / confident
D.2	Agreeableness	A measure of one's trustful and helping nature.	• Friendly / compassionate • Analytic / detached
D.3	Extraversion	Quite generally, this describes the tendency to seek social interaction.	• Outgoing / energetic • Solitary / reserved
D.4	Conscientiousness	Planned rather than spontaneous behaviour.	• Efficient / organized • Easy going / careless
D.5	Openness	Reflects the degree of intellectual curiosity.	• Inventive / curious • Consistent / cautious

Table 10. Psychological traits

#	Name	Description	Example values and range
E.1	Sexual preference	Sexual preference.	• Heterosexual • Homosexual • Bisexual

Table 11. Sexual profile

#	Name	Description	Example values and range
F.1	Parties	We will have separate lists per country.	Part of list for Belgium: • CD&V • Groen! • N-VA • Open VLD • etc. Part of list for Sweden: • Centerpartiet • Vansterpartiet • Folkpartiet liberalerna • etc.
F.2	Political ideology	This is clearly correlated to the above; however, we included it in order to take into account the case that the specific party that the user supports is not listed.	• Communist • Socialist • Green • Liberal • Christian democratic • Conservative • Right-wing extremist

Table 12. Political attitudes

#	Name	Description	Example values and range
G.1	Supported religion	A finer division may eventually be used.	• Atheist • Agnostic • Christian • Muslim • Hinduist • Buddhist • Other

Table 13. Religious beliefs

#	Name	Description	Example variables and range
H.1	Smoking	We may eventually use finer classes that will also reflect the frequency of smoking	• Smoker • Non-smoker
H.2	Drinking (alcohol)	Alcohol consumption.	• Non-drinker • Social drinker • Drinker • Alcoholic
H.3	Drug use	This includes all types of substances that could be classified as drugs	• Yes • No
H.4	Chronic diseases	The list on the right column is only indicative. Can take multiple values simultaneously.	• Diabetes • Epilepsy • Cardiomiopathy • Hypertension • etc.
H.5	Disabilities	Different kinds of disabilities (physical, mental, sensory, etc.). The list on the right column is only indicative.	• Lower limb loss • Vision impairment • Balance disorder • etc.
H.6	Other health factors	Additional health factors. Likely to be extended. Can take multiple values simultaneously.	• Exercise (yes / no) • Late night shifts (yes / no) • Staying up late

Table 14. Health Factors and Condition

#	Name	Description	Example values and range
I.1	Home	The place where the user resides. Different levels of accuracy are possible.	City, address, country, GPS
I.2	Work	The place where the user works. Different levels of accuracy are possible.	City, address, country, GPS
I.3	Favourite places	Favourite places of the user. Different levels of accuracy are possible. Can take multiple values simultaneously.	City, address, country, GPS
I.4	Visited places	Places that the user has visited. Different levels of accuracy are possible. Can take multiple values simultaneously.	City, address, country, GPS

Table 15. Location

#	Name	Description	Example values / range
J.1	Brand attitude	Brand + stance (favourable, non-favourable). Can take multiple values simultaneously.	• List of pairs, each of which has a brand name and either the value “favourable” or the value “non-favourable”
J.3	Hobbies	Hobbies. Can take multiple values simultaneously.	A list of hobbies
J.4	Devices	Used electronic devices. Can take multiple values simultaneously.	• Smartphone • Tablet • PC • etc.

Table 16. Consumer Profile

Annex 2 – Prototype Implementations

This document is accompanied by a number of prototype implementations. Details about each of them are provided in the following.

Disclosure scoring framework

The code for the implementation of the disclosure scoring framework has been made publicly available at this location:

https://github.com/MKLab-ITI/usemp-pscore

As mentioned, the framework has been implemented in Python (version 3.4 is required). Additionally, the following modules are required:

- aniso8601==0.92
- Flask==0.10.1
- Flask-RESTful==0.3.2
- flask-restful-swagger
- gunicorn==19.3.0
- itsdangerous==0.24
- Jinja2==2.7.3
- MarkupSafe==0.23
- pymongo==2.8
- pytz==2015.2
- six==1.9.0
- virtualenv==12.1.1
- Werkzeug==0.10.4
- newrelic
- flask-cors

To start using the code, please execute the script in the file scoringFramework.py. For more details please see the aforementioned pages.

Disclosure settings framework

The privacy settings implementation, as described in Chapter 4 is provided together with this deliverable. The code has been written in Java. A complete Netbeans project, including the source code, can be found in the folder privacy_settings_assistance. Apart from the code the following sample files are provided for demo purposes:

- policy_sample.json. This contains the very simple sample policy that was listed in Chapter 4. Importantly though, it contains all different types of content and audience definitions and can be used as an example based on which quite complicated policies can be built.
- policy_sample_simplified.json. This file contains the simplified version of the sample policy that is contained in policy_sample.json.
- users_sample.json. This file contains an example set of friends together with their properties. This is used from the demo code and can be easily adapted for experimentation.
- content_sample.json. This file contains an example set of content posted by some user. Just like users_sample.json, this is only used from the demo code.

The demo can be executed by running the main function that can be found in the Main package. The user can select if the full or the simplified policy will be used. In both cases, the code will load the set of users and content from the respective files, it will also load the appropriate policy file and will produce a set of sharing recommendations for each item that is included in the file content_sample.json. It is quite straightforward to adapt the policy in the json file and examine how the recommendations change as the policy changes.

Please see the README file in the provided package for more details.

Automatic detection of social circles

The data required to reproduce the results that have been presented are provided by the authors of (Leskovec and McAulley, 2012):

- https://snap.stanford.edu/data/egonets-Facebook.html (Facebook)
- https://snap.stanford.edu/data/egonets-Twitter.html (Twitter)

The code that is provided with this deliverable is written in Java and can be found in the folder “egonetwork_analysis”. To run it, please execute the class with the name EgonetworkAnalysis. The code will run in batch mode all the experiments for either the Facebook or Twitter dataset depending on the options selected by the user. Please also note that the user has the option to also call an evaluation script that we also provide and that is based on code also provided by (Leskovec and McAulley, 2012). To run the evaluation script a cygwin installation is required. For more details please see the README file in the provided package.

Annex 3 – Mahout Evaluation Tests

The following snippets of code were used to carry out the tests of Chapters 4 and 5.

For evaluation, the GenericRecommenderIRStatsEvaluator[10] was used:

```
RecommenderIRStatsEvaluator eval = new GenericRecommenderIRStatsEvaluator();
            IRStatistics stats = eval.evaluate(
                  builder, //Recommender builder
                  null, // datamodel builder
                  model, //dataset
                  null, IDRescorer
                  i, //recommendations at
                  GenericRecommenderIRStatsEvaluator.CHOOSE_THRESHOLD, //relevanceThreshold
                  0.9 // evaluation percentage
);
```

The builder variable is the recommendation builder engine we used. For all the experiments, the following six instances were used.

```
   static class LogLikeItemRecommenderBuilder implements RecommenderBuilder{
      public Recommender buildRecommender(DataModel dataModel) throws TasteException {
         ItemSimilarity similarity = new LogLikelihoodSimilarity(dataModel);
//          return new GenericBooleanPrefItemBasedRecommender(dataModel, similarity);
         return new GenericItemBasedRecommender(dataModel, similarity);
      }
   }
   static class EuclideanItemRecommenderBuilder implements RecommenderBuilder{
      public Recommender buildRecommender(DataModel dataModel) throws TasteException {
         ItemSimilarity similarity = new EuclideanDistanceSimilarity(dataModel);
//          return new GenericBooleanPrefItemBasedRecommender(dataModel, similarity);
         return new GenericItemBasedRecommender(dataModel, similarity);
      }
   }

   static class TanimotoItemRecommenderBuilder implements RecommenderBuilder{
      public Recommender buildRecommender(DataModel dataModel) throws TasteException {
         ItemSimilarity similarity = new TanimotoCoefficientSimilarity(dataModel);
//          return new GenericBooleanPrefItemBasedRecommender(dataModel, similarity);
         return new GenericItemBasedRecommender(dataModel, similarity);
      }
   }
   static class TanimotoUserRecommenderBuilder implements RecommenderBuilder{
      public Recommender buildRecommender(DataModel dataModel) throws TasteException {
         UserSimilarity similarity = new TanimotoCoefficientSimilarity(dataModel);
```

[10] http://archive.cloudera.com/cdh4/cdh/4/mahout-0.7-cdh4.2.2/mahout-core/org/apache/mahout/cf/taste/impl/eval/GenericRecommenderIRStatsEvaluator.html

```
            UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity, dataModel);
            return new GenericUserBasedRecommender(dataModel,neighborhood, similarity);
//            return new GenericBooleanPrefUserBasedRecommender(dataModel,neighborhood, similarity);
        }
    }

    static class EuclideanUserRecommenderBuilder implements RecommenderBuilder{
        public Recommender buildRecommender(DataModel dataModel) throws TasteException {
            UserSimilarity similarity = new EuclideanDistanceSimilarity(dataModel);
            UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity, dataModel);
            return new GenericUserBasedRecommender(dataModel,neighborhood, similarity);
//            return new GenericBooleanPrefUserBasedRecommender(dataModel,neighborhood, similarity);
        }
    }

    static class LogLikeUserRecommenderBuilder implements RecommenderBuilder{
        public Recommender buildRecommender(DataModel dataModel) throws TasteException {
            UserSimilarity similarity = new LogLikelihoodSimilarity(dataModel);
            UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.1, similarity, dataModel);
            return new GenericUserBasedRecommender(dataModel,neighborhood, similarity);
//            return new GenericBooleanPrefUserBasedRecommender(dataModel,neighborhood, similarity);
        }
    });
```

References

Acquisti, C. M. Fong. (2012). An Experiment in Hiring Discrimination Via Online Social Networks. Social Science Research Network Working Paper Series, Apr. 2012.

Balasubramanyan, R., Cohen, W. Block-LDA: Jointly modelling entity-annotated text and entity-entity links, In ICML 2010 Workshop on Topic Models: Structure, Applications, Evaluation, and Extensions, 2010.

Blei, D., Ng, A., Jordan, M. Latent Dirichlet Allocation, Journal of Machine Learning Research 3 (March 2003), 993-1022, 2003.

Chang, J., Boyd-Graber, J. Relational topic models for document networks, In Proceedings of the 12th International Conference on Artificial Intelligence and Statistics AISTATS, 2009.

Dev, H., Ali, M., Hashem, T. User Interaction Based Community Detection in Online Social Networks, Database Systems for Advanced Applications. LNCS volume 8422, 296-310. 2014.

Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874

Fogues, R., Such, J., Espinosa, A. BFF: A tool for eliciting tie strength and user communities in social networking services, Information Systems Frontiers 16.2, 225-237, 2014.

Freund, Y., Schapire, R. Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, San Francisco, 148-156, 1996

Gilbert, E., Karahalios, K. Predicting Tie Strength with Social Media, In Proceedings of the 27th International Conference on Human Factors in Computing Systems, 2009.

Jia, Y. Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding. http://caffe.berkeleyvision.org, 2013.

Jones, S., O'Neill, E. Feasibility of structural network clustering for group-based privacy control in social networks. Proceedings of the 6th Symposium on Usable Privacy and Security, 2010.

Kelley, P., Brewer, R., Mayer, Y., Cranor, L., Sadeh, N. An Investigation into Facebook Friend Grouping. Human-Computer Interaction INTERACT 2011. Lecture Notes in Computer Science Volume 6948, pp. 216-233, 2011.

Knijneburg, B.P., Koba, A., Jin, H.: Dimensionality of information disclosure behavior. International Journal of Human-Computer Studies. 71(12), 1144-1162, 2013.

Knijneburg, B.P.: Information Disclosure Profiles for Segmentation and Recommendation. In Symposium on Usable Privacy and Security (SOUPS), 2014.

Leskovec, J., McAuley, J. Learning to Discover Social Circles in Ego Networks, Advances in Neural Information Processing Systems 25, 2012.

Li, J.. Fine, J. Weighted Area Under the Receiver Operating Characteristic Curve and Its Application to Gene Selection. Journal of the Royal Statistical Society Series C, Applied statistics. 2010;59(4):673-692. doi:10.1111/j.1467-9876.2010.00713.x.

Liu, Y., Niculescu-Mizil, A., Gryc, W. Topic-link LDA: joint models of topic and author community. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). ACM, New York, NY, USA, 665-672, 2009.

Madejski, M., Johnson, M., Bellovin, S. A study of privacy settings errors in an online social network, 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), IEEE, 2012.

Minka, T., Lafferty, J. Expectation-propagation for the generative aspect model, In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, 2002.

Newman, M., Girvan, M. Finding and evaluating community structure in networks, Physical review E 69, no. 2, 2004.

Petkos, G., Papadopoulos, S., Kompatsiaris, Y. (2015a). PScore: a framework for enhancing privacy awareness in online social networks. MFSec 2015.

Petkos, G., Papadopoulos, S., Kompatsiaris, Y. (2015b). Social Circle Discovery in Ego-Networks by Mining the Latent Structure of User Connections and Profile Attributes. ASONAM 2015.

Read, J., Pfahringer, B., Holmes, G. Multi-label classification using ensembles of pruned sets. Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. IEEE, 2008.

Ruan, Y., Fuhry, D., Parthasarathy, S. Efficient community detection in large networks using content and links, In Proceedings of the 22nd international conference on World Wide Web (WWW ’13), 1089-1098, 2013.

Spiliotopoulos, T., Pereira, D., Oakley, I. Predicting Tie Strength with the Facebook API, Proceedings of the 18th Panhellenic Conference on Informatics, 1-5, 2014.

Spyromitros-Xioufis, E. et al. Multi-label classification methods for multi-target regression. arXiv preprint arXiv:1211.6581, 2012.

Streich, A., Frank, M., Basin, D., Buhmann, J. Multi-asignment clustering for boolean data, Journal of Machine Learning Research 13, no. 1, 459-489, 2012.

Tsoumakas, G., Katakis, I., Vlahavas, I. Mining multi-label data.Data mining and knowledge discovery handbook. Springer US, 2010. 667-685.

Xu, X., Yuruk, N., Feng, Z., Schweiger, T. SCAN: a structural clustering algorithm for networks, In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD ’07). ACM, New York, NY, USA, 824-833, 2007.

Yang, J., McAuley, J., Leskovec, J. Community Detection in Networks with Node Attributes, ICDM ’13, 2013.

Yoshida, T. Towards finding hidden communities based on user profiles, In ICDM Workshops, 2010.